electronics

MDPI

Article

A Certificateless Verifiable Bilinear Pair-Free Conjunctive Keyword Search Encryption Scheme for IoMT

Weifeng Long [1,2,*], Jiwen Zeng [1], Yaying Wu [3], Yan Gao [2] and Hui Zhang [2]

[1] School of Mathematical Sciences, Xiamen University, Xiamen 361005, China; jwzeng@xmu.edu.cn
[2] School of Mathematical Sciences, Guizhou Normal University, Gui'an New District, Guiyang 550001, China; 19010060164@gznu.edu.cn (Y.G.); 21020061214@gznu.edu.cn (H.Z.)
[3] School of Big Data and Computer Science, Guizhou Normal University, Gui'an New District, Guiyang 550001, China; 21010230690@gznu.edu.cn
* Correspondence: 460143769@gznu.edu.cn

Abstract: With superior computing power and efficient data collection capability, Internet of Medical Things (IoMT) significantly improves the accuracy and convenience of medical work. As most communications are over open networks, it is critical to encrypt data to ensure confidentiality before uploading them to cloud storage servers (CSSs). Public key encryption with keyword search (PEKS) allows users to search for specific keywords in ciphertext and plays an essential role in IoMT. However, PEKS still has the following problems: 1. As a semi-trusted third party, the CSSs may provide wrong search results to save computing and bandwidth resources. 2. Single-keyword searches often produce many irrelevant results, which is undoubtedly a waste of computing and bandwidth resources. 3. Most PEKS schemes rely on bilinear pairings, resulting in computational inefficiencies. 4. Public key infrastructure (PKI)-based or identity-based PEKS schemes face the problem of certificate management or key escrow. 5. Most PEKS schemes are vulnerable to offline keyword guessing attacks, online keyword guessing attacks, and insider keyword guessing attacks. We present a certificateless verifiable and pairing-free conjunctive public keyword searchable encryption (CLVPFC-PEKS) scheme. An efficiency analysis shows that the performance advantage of the new scheme is far superior to that of the existing scheme. More importantly, we provide proof of security under the standard model (SM) to ensure the reliability of the scheme in practical applications.

Keywords: Internet of Medical Things; IoMT; certificateless encyption with keyword search; standard model; offline keyword guessing attacks; online keyword guessing attacks; insider keyword guessing attacks; security; privacy

Citation: Long, W.; Zeng, J.; Wu, Y.; Gao, Y.; Zhang, H. A Certificateless Verifiable Bilinear Pair-Free Conjunctive Keyword Search Encryption Scheme for IoMT. *Electronics* **2024**, *13*, 1449. https://doi.org/10.3390/electronics13081449

Academic Editor: Baris Aksanli

Received: 30 January 2024
Revised: 4 April 2024
Accepted: 7 April 2024
Published: 11 April 2024

1. Introduction

The Internet of Things can connect any item to the Internet. It uses information-sensing devices such as radio frequency identification and infrared sensors to transmit data and communicate according to specific protocols, thus realizing intelligent identification, positioning, tracking, monitoring, and management functions [1–3]. As shown in Figure 1, as an application of Internet of Things technology in the medical field, IoMT [4] can closely connect medical staff, patients, and various medical devices to achieve real-time feedback on patient health status. It not only improves the speed of medical response and provides all-weather medical care but also alleviates the workload of medical staff. IoMT improves the precision and convenience of medical work, leading to a better quality of medical care. IoMT also plays a significant role in saving lives, helping to control costs, and improving efficiency.

Electronic medical records (EMRs) [5,6] play a crucial role in IoMT. With the acceleration of the digitization of medical data, the amount of EMR data has increased rapidly. Storing and managing EMRs has become a significant challenge. Fortunately, cloud computing technology offers a solution to this challenge. Hospitals can store EMRs in the cloud,

eliminating the cost of local management and maintenance and enabling efficient data sharing and utilization.

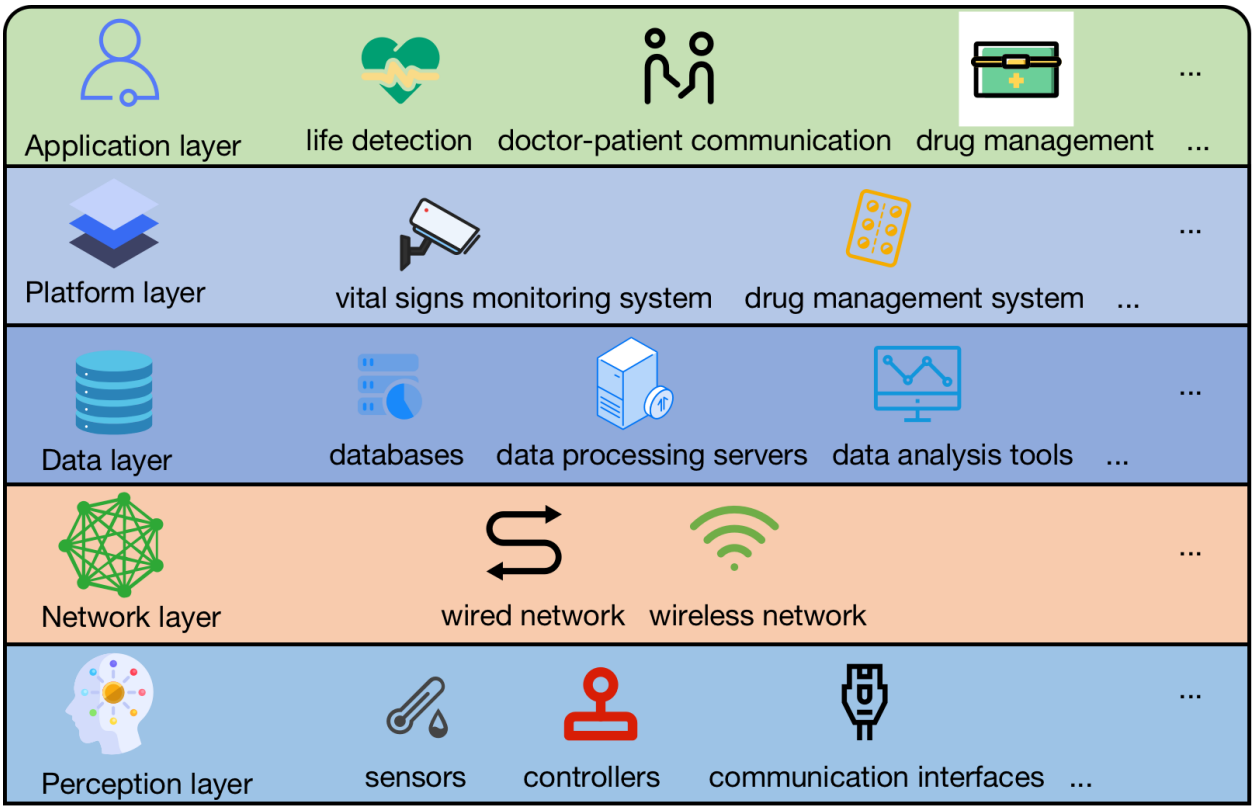

Figure 1. System model of IoMT.

Because EMRs contain private patient information, they often need to be encrypted to ensure patient privacy. However, this encryption method makes it inconvenient for users to search for specific keywords in EMRs. The simplest solution is for users to download all the ciphertext data, decrypt them individually, and then search among them. This approach is impractical. It leads to exceedingly high computational and communication costs. To solve this problem, researchers have proposed searchable encryption (SE) [7] technology. SE is an encryption primitive that allows users to perform keyword searches on encrypted data.

In reality, we are more likely to encounter multi-data owner scenarios, such as a patient's EMR, that often need to be co-managed by multiple doctors, departments, or healthcare organizations. If each party independently owns some of the data in an EMR, this scenario is a non-shared multi-owner scenario [8]. If, on the other hand, multiple parties jointly own the data of an EMR, then this scenario is a shared multi-owner scenario [9]. In a non-shared multi-owner scenario, each staff member is responsible for a specific portion of the EMR. Each part of the data is a separate record, requiring independent computation of indexes and signatures. This processing will result in a linear increase in the cost of storing indexes and signatures as the number of owners increases, as well as an increase in verification and encryption time, thus significantly increasing the overall computation and storage costs. In contrast, the shared multi-owner scenario allows multiple owners to sign on the same EMR, ultimately generating a multi-signature. The size of the multi-signature is constant and independent of the number of owners. Therefore, the verification time and storage overhead of signatures are independent of the number of owners. For indexing in the shared multi-owner setup, the whole EMR is given only a single index through which the retriever can search the multi-owner EMR. In this way, the shared multi-owner significantly saves time and space costs, which makes it more advantageous in real-world applications. In recent years, some research works have also verified the ad-

vantages of the shared multi-owner setting. Miao et al. [10] proposed a keyword-searchable encryption scheme with a hidden access policy under the shared multi-owner setting. Padhya et al. [11] proposed a new key aggregation-searchable encryption scheme supporting sorted queries on encrypted datasets and a multi-keyword multidimensional search on multi-owner datasets. However, most PEKS solutions deployed in shared multi-owner scenarios face significant computational and storage costs.

The current PEKS scheme leaves much to be desired in terms of efficiency and security: PKI-based or identity-based cryptosystems encounter certificate management and key escrow problems during system deployment; the use of secure channels leads to inefficiency in the PEKS scheme; single-keyword searches inevitably produce many irrelevant results, leading to a waste of bandwidth and computational resources; CSS, as a semi-trusted third party, may provide incorrect search results to save computational and bandwidth resources; a large number of pairing operations puts a heavy computational burden on the system; the PEKS scheme is vulnerable to offline keyword-guessing attacks, online keyword-guessing attacks, and internal keyword-guessing attacks; most of the searchable encryption schemes are proven to be secure in the random oracle model (ROM) but the ROM is not suitable in the idealized model, resulting in security that cannot be fully guaranteed in practical applications; most searchable encryption schemes incur significant computational and storage costs when deploying PEKS schemes in non-shared multi-owner setups. In reality, it is necessary to solve the above problems, which provides a new direction for our research.

1.1. Our Contribution

We construct a certificateless-based, verifiable, pairing-free, conjunctive keyword search encryption scheme (CLVPFC-PEKS). Specifically, the main contributions are as follows:

- Conjunctive multi-keyword search: The new solution allows users to search for multiple keywords without increasing the number of trapdoors and ciphertexts, significantly improving the accuracy of search results.
- Verifiability of the search results: The new scheme attaches a signature to each document. The signature will then be used to verify the search results, ensuring the accuracy of the search results and preventing users from wasting time and resources on invalid results.
- Certificateless-based: The new scheme overcomes the limitations of certificate management and key escrow in existing searchable encryption schemes.
- No pairing: Pairing operations take time, so the computational efficiency can be significantly improved by not using pairing operations.
- No need for a secure channel: The new solution eliminates the need for a secure channel and reduces system construction costs.
- Shared multi-owner settings: The new scheme allows users to search for document sets shared by multiple users using a single trapdoor.
- Proven security under the standard model: The new scheme is secure against offline keyword guessing attacks and chosen keyword attacks (CKAs) in the standard model. Meanwhile, based on the security of the Diffie–Hellman shared secret keys, the new scheme can also resist online keyword guessing attacks (e.g., file injection attack (FIA)) and insider keyword guessing attacks (IKGAs).

In Table 1, we compare the features of PKES schemes. Currently, no SE scheme can provide result verification, can provide a conjunctive keyword search, is secure channel-free, is certificateless-based, is pairing-free, and can provide support for shared multi-owners simultaneously. In particular, none of the pairing-free SE schemes support result verification and conjunctive keyword searches.

Table 1. Functional comparison.

Scheme	SRV	CKS	CL	SCF	PF	SMO
[12]	×	√	×	×	×	×
VCSE [13]	√	√	×	√	×	×
VCKSM [14]	√	√	×	√	×	√
VMKDO [15]	√	√	×	×	×	×
VMKS [16]	√	√	√	×	×	×
mCLPECK [17]	×	√	√	×	×	×
CL-dPAEKS [18]	×	×	√	√	×	×
[19]	×	√	×	×	×	×
[20]	×	×	√	√	√	×
[21]	×	×	√	√	√	×
[22]	×	×	√	√	√	×
ours	√	√	√	√	√	√

SRV: verifiability of search results. CKS: conjunctive keyword search. CL: certificateless-based. SCF: secure-chanel free. PF: pairing-free. SMO: shared multi-owner. ×: not supporting this functionality. √: supporting this functionality.

1.2. Organization

The following is the framework for the rest of this article. We summarize some related work in Section 2. We discuss preparatory knowledge in Section 3. We define the system model and the security model of the scheme in Section 4. We show the details of the CLVPFC-PEKS scheme in Section 5. We discuss the security of the CLVPFC-PEKS scheme in Section 6. We analyze the effectiveness of CLVPFC-PEKS in Section 7. Finally, we summarize this paper in Section 8.

2. Related Work

In 2004, Boneh et al. [23] first proposed the concept and construction of public key encryption with keyword search(PEKS), which laid the foundation for research in this field. However, the early PEKS schemes still had many shortcomings in efficiency and safety. Baek [24] points out that the PEKS scheme of [23] is inefficient because [23] uses safe channels. To eliminate the requirement for secure channels, Baek et al. [24] proposed PEKS without secure channels (SCF-PEKS), also known as PEKS with designated servers/testers (dPEKS) [25]. However, most SCF-PEKS schemes are PKI-based or ID-based, which inevitably incurs problems with certificate management and key escrow. To overcome these challenges, Peng et al. [18], based on the certificateless cryptosystem [26], proposed a keyword-searchable encryption scheme (CL-SCF-PEKS) without a secure channel. Since then, many papers have studied certificateless searchable encryption schemes, which inject new vitality into the development of this field. It is worth noting that most PEKS schemes [18,27–30] search for a single keyword. However, this single-keyword search method often produces irrelevant results, wasting bandwidth and computing resources. Golle et al. [31] proposed the first conjunctive keyword-searchable encryption scheme to address the issue of resource waste. Subsequently, more searchable encryption schemes that support conjunctive keywords have been proposed in recent years [14,32,33].

Given that the computational complexity of pairing is much higher than that of scalar multiplication on the group of elliptic curves, designing schemes that do not require pairing will significantly improve the computational efficiency of the schemes. Current SE schemes that do not require pairing operations are still scarce, and these schemes [21,22,34–40] remain to be improved. In 2019, Lu et al. [39] proposed a PEKS scheme based on the certificateless cryptosystem, which is pairing-free, and proved the indistinguishability of keyword ciphertexts (CL-PF-PEKS). However, Ma et al. [40] showed that the work

of [39] is insecure under user-simulated attacks and provided a new CL-PF-PEKS scheme. Recently, [21,22] proposed CL-PF-PEKS schemes that are both secure and effective, respectively, but unfortunately, they are single-keyword-searchable.

Existing SCF-PEKS schemes have significant shortcomings in terms of security, especially their vulnerability to offline keyword guessing attacks [41], online keyword attacks (e.g., FIA) [19,42], and internal keyword attacks [43]. PEKS schemes typically use keywords from low-entropy spaces, making them vulnerable to offline keyword-guessing attacks. In this attack, both internal and external attackers can use a testing algorithm to accurately guess the keywords corresponding to the given keyword trapdoors. The attack works because the generation of trapdoors uses only keywords and public keys. Currently, the offline keyword guessing attack has become one of the most destructive attacks on PEKS schemes because it can lead to the leakage of encrypted data and information related to the keywords contained in the trapdoor. Unfortunately, Yang et al. [44] showed that even after modifying the trapdoor structure of PEKS schemes, an attacker can still perform an online keyword guessing attack(e.g., FIA). Unlike external offline keyword attacks, external online keyword guessing attacks allow attackers to guess keywords by analyzing search results in the cloud in real time. Specifically, the external attacker first generates keyword ciphers and data ciphers containing all possible keywords and data ciphers using the public keys of the cloud server and the retriever and injects these ciphers into the cloud server. Subsequently, the external attacker monitors the communication between the cloud server and the retriever. Once the attacker finds that the search results returned by the server match the previously injected ciphertexts, they can determine the keywords that the retriever is searching for. In addition, the PEKS scheme is also vulnerable to internal malicious CSS attacks. Jeong et al. [45] demonstrated that the PEKS framework is susceptible to internal offline keyword attacks initiated by malicious CSSs. Wang et al. [46] pointed out that even though the trapdoor is indistinguishable, the SCF-PEKS scheme is still not able to defend against keyword-guessing attacks by malicious CSSs. More seriously, Shao et al. [47] proposed a generic attack method and pointed out that it is almost impossible to construct a SCF-PEKS scheme that can defend against malicious CSSs. This is because malicious CSSs can perform keyword encryption and testing algorithms, making the SCF-PEKS scheme particularly vulnerable against offline keyword-guessing attacks from malicious internal servers.

In addition, CSSs, as semi-trusted entities, may selectively perform a few search operations and return some inaccurate search results to save computational and bandwidth resources. Therefore, PEKS schemes should provide an authentication mechanism to ensure the accuracy of search results without decryption. Reference [48] proposed the first keyword-searchable encryption scheme that supports authentication. Since then, numerous researchers have explored keyword-searchable encryption schemes that support authentication [15,16,49]. The authors of [16] proposed a verifiable multiple keywords search(VMKS) scheme, claiming that it resists keyword guessing attacks under the standard model.

3. Preliminaries

Let $q > 3$ be a large prime, F_q be a prime field, and the elliptic curve E over the field F_q satisfy the equation $y^2 \ mod \ q = (x^3 + ax + b) \ mod \ q$, where $a, b, x, y \in F_q$ and $(4a^3 + 27b^2) \ mod q \neq 0$. All points on E and the infinite point O form a cyclic group. The elliptic curve cryptosystem (ECC) has the following difficulties:

- The elliptic curve discrete logarithm problem (ECDLP): given $P, Q \in G_q$, where P is the generator of the group and Q is the element in G_q. It is difficult to calculate the integer k, such that $Q = kP$, where $k \in Z_{q^*}$.
- The elliptic curve computational Diffie–Hellman problem (ECDCHP): for any given a, b in Z_{q^*}, it is difficult to calculate abP, where $(P, aP, bP) \in G_q$.
- The elliptic curve decisional Diffie–Hellman (ECDDDH) problem: given $aP, bP \in G$, where a, b are unknown. The decisional Diffie–Hellman (DDH) problem is to decide whether X equals abP or a random element in G.

4. The System Model and Security Model of CLVPFC-PEKS

4.1. System Model

There are five core entities involved in the new scheme: the key generation center (KGC), the data owners (DOs, including patients and their doctors), CSSs, retrievers (e.g., authorized doctors or hospitals), and a private audit server (PAS), as shown in Figure 2.

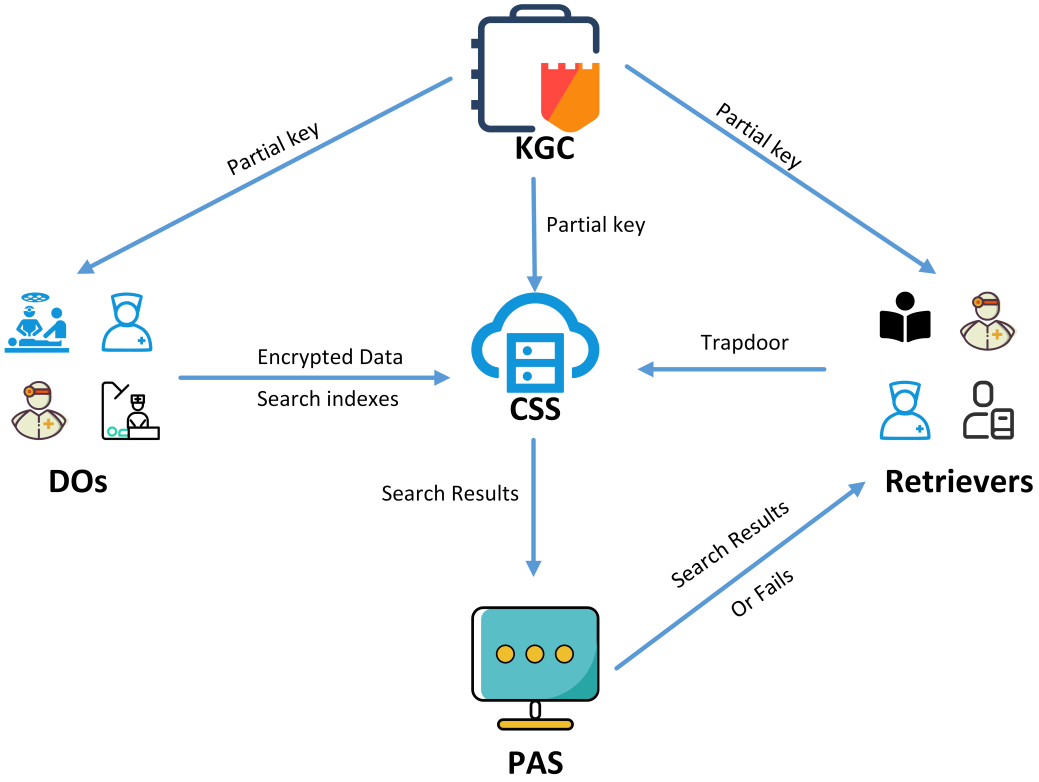

Figure 2. System model of CLVPFC-PEKS.

- The KGC is responsible for generating the public parameters of the system and some of the private keys for the DOs, the CSSs, and the retrievers.
- DOs have many files to store and manage, but their resources and capabilities are limited. From the perspective of resource-saving and data security, the data owner chooses to upload the encrypted data files and indexes (keyword ciphertext) to the cloud storage server.
- The CSS, as a semi-trusted entity with powerful computing and storage capabilities, provides data storage and retrieval services to authorized cloud customers. When a retriever initiates a retrieval request, the cloud storage server looks up the keyword traps and returns the corresponding data files to the retriever.
- Retrievers can initiate a ciphertext retrieval request by sending a keyword trapdoor to the cloud storage server and decrypt the received ciphertext to obtain the required information.
- The PAS, as a fully trusted entity, is responsible for validating the search results to ensure that the data files received by the searcher are accurate.

The CSS is semi-trustworthy and curious. It may selectively perform a few search operations and provide some erroneous search results to conserve its resources. At the same time, CSSs try to snoop on valuable and sensitive information. The PAS is fully trusted to ensure the accuracy of search results. In addition, authorized retrievers can confidently initiate search requests without worrying about leaking valuable information to CSSs.

4.2. Solution Framework

To better understand the notations in our proposed scheme, Table 2 explains the pre-defined notations used throughout this paper. We set integer k as the security level and (F, W) as the EMR file and the keyword set contained in EMR.

Table 2. Notation descriptions.

Notations	Descriptions
x	Master secret key
P_{pub}	System public key
$O = \{O_1, O_2, \cdots, O_d\}$	Data owner collection
$ID_i \in \{0,1\}^* (1 \leqslant i \leqslant d)$	Identity set for data owner O_i
ID_C	Identity for CSS
ID_u	Identity for data user
(PK_C, SK_C)	Public/secret key pair for CSS
(PK_{O_i}, SK_{O_i})	Public/secret key pair for data owner O_i
(PK_u, SK_u)	Public/secret key pair for data user
$F = \{f_1, f_2, \cdots, f_n\}$	File set F
$C = \{c_1, c_2, \cdots, c_n\}$	Ciphertext set
$ID = \{id_1, id_2, \cdots, id_n\}$	Identity set for F
$W_i = \{w_{i1}, w_{i2}, \cdots, w_{im}\}$	Collection of keywords
$sig_{i,t}$	Data owner O_i signature for c_t
sig_t	Data owners' multi-signature for c_t
$Sig = \{sig_1, sig_2, \cdots, sig_n\}$	F's multi-signature
I_i	Index of f_i
$I = \{I_1, I_2, \cdots, I_n\}$	Index set for F
$W' = \{w'_1, w'_2, \cdots, w'_l\}$	Search keyword set
$T_{W'}$	Trapdoor of W'
$C' = \{c_{k_1}, c_{k_2}, \cdots, c_{k_s}\}$	Search results
$ID' = \{id_{k_1}, id_{k_2}, \cdots, id_{k_s}\}$	Identity set for C'

Definition 1 (CLVPFC-PEKS). *Our scheme is a tuple of six algorithms, as follows:*

SetUp(1^k)*: Given the security parameter k, the KGC outputs the public parameters* Ω*, the system public key* P_{pub}*, and the master secret key msk for the traditional public key algorithm.*

KeyGen (Ω, O, U, CSS)*: For the data owner set O, the data user U, and the CSS, the KGC generates the public/secret key pairs* $\{PK_{O_i}, SK_{O_i}\}(1 \leqslant i \leqslant d)$*,*$\{PK_u, SK_u\}$*, and* $\{PK_C, SK_C\}$*, respectively.*

Set-Secret-Value: After inputting the public parameters Ω*, this probabilistic algorithm outputs data owners, data users, and CSS's Secret-Value.*

Partial-Private-Key-Extract: The KGC executes this algorithm, which accepts the identity of the data owner, data user, and CSS, then uses them in combination with the master key to generate a partial private key for the data owner, data user, and CSS.

Set-Private-Key: Set the full secret keys of the data owner, data user, and CSS.

Set-Public-Key: Set the full public keys of the data owner, data user, and CSS.

$Enc(\Omega, F, W, \{ID_i\}, ID, \{SK_{O_i}\}, \{PK_{O_i}\}, PK_u)$*: DOs first conduct this probabilistic algorithm to generate the ciphertext set C for the set F. Then, DOs generate multiple signatures Sig and index set I for ciphertext C. Then, they send the tuple* (C, I, Sig) *to the CSS.* $TraGen(\Omega, SK_u, W')$*; given the keyword* W'*, the DO runs this algorithm to output trapdoor* $T_{W'}$ *.*

Test$(\Omega, T_{W'}, I)$*: Using the trapdoor* $T_{W'}$ *as an input, the CSS matches it with the index set* I*, then returns the relevant ciphertext* $C' \subseteq C$ *and signature* Sig' *set to PAS.*

Verify $(\Omega, C', ID', sig, \{PK_{O_i}\})$*: The PAS runs this algorithm by initiating interaction with the CSS to check the correctness of the search result* $C_{W'}$*. If* $C_{W'}$ *passes the result validation, the PAS will return it to the retriever. Otherwise, it will abort the algorithm.*

4.3. Security Model

Our scheme primarily considers security in the following two aspects: (1) The index corresponding to the keyword has security against choice keyword attacks. (2) The trapdoor used for queries should possess security.

Because our scheme is certificateless-based, there are two different classes of adversaries: A_1 and A_2. Therefore, when discussing the security of the index and trapdoor, we must analyze and prove them from the angles of these two adversaries.

A_1: A_1 does not know the master key, but A_1 can replace any user's public key.

A_2: A_2 knows the master key, but A_2 cannot replace any user's public key.

We call these adversaries the adversary of type-1 and the adversary of type-2.

Mahmoud Ismail et al. [50] articulated the basic principles of the zero-trust framework, delved into the threat landscape facing IoT systems, and assessed how the zero-trust principle can effectively address these threats. Jamal A. Alenizi et al. [51] investigated various risks and vulnerabilities that may affect the operation of blockchain-based smart healthcare systems, especially ransomware, and proposed a framework for mitigating healthcare ransomware attacks. The framework proposed by Jamal A. Alenizi et al. has higher computational efficiency and lower communication overhead than similar existing frameworks.

4.3.1. Ciphertext Indistinguishability against Chosen Keyword Ciphertext Attack

A PEKS scheme with ciphertext indistinguishability against chosen keyword and ciphertext attacks (CKCA-CIND) feature can protect the data owner's keyword ciphertext stored in the cloud from revealing relevant keyword information. When encrypted data are stored in the CSS, it will attach the corresponding keywords $\{w_{i1}, w_{i2}, \cdots, w_{im}\}$. Even if the keyword ciphertext is captured during transmission, no adversary can obtain keywords embedded in the keyword ciphertext.

We will define the definition of CKCA-CIND. There are two games to discuss the security of CKCA-CIND.

Game 1.

A_1 simulates malicious users, and B is the challenger. B and A_1 play this game together.

Setup: B runs the SetUp(1^k) program to obtain Ω, P_{pub} and msk. B sends P_{pub} to A_1 and keeps msk. Then, B sets the key pair of O_i $(i \in \{1, 2, \cdots, d\})$ and CSS, i.e., (PK_{O_i}, SK_{O_i}) $(i \in \{1, 2, \cdots, d\})$ and (PK_c, SK_c). B sends the public keys PK_{O_i} $(i \in \{1, 2, \cdots, d\})$ and PK_c to A_1, while SK_{O_i} and SK_c are unknown to A_1.

Phase 1. A_1 executes the User-Public-Key query before executing other queries. It sets up lists to store the above queries and answers. All lists are initially empty. A_1 makes the queries to challenger B as follows:

- User-Public-Key query: When A_1 inputs the identity ID_u, B outputs the user's public key PK_u.
- Replace-Public-Key query: A_1 inputs (ID_j, PK_j'), B replaces PK_j with PK_j'.
- Secret-Value query: When A_1 inputs the identity ID_j, B returns the secret value corresponding to the ID_j. If PK_j is replaced, B refuses to answer.
- Partial-Private-Key-Extract query: When A_1 enters the ID_j, if $ID_j = ID_u^{\diamondsuit}$($ID_u^{\diamondsuit}$ is the challenge identity), B fails and stops. Otherwise, B returns the corresponding Partial-Private-Key.
- Keyword Ciphertext Query: A_1 asks B for the keyword ciphertext of any keyword W it cares about. B runs the $Enc(\Omega, F, W, \{ID_i\}, ID, \{SK_{O_i}\}, \{PK_{O_i}\}, PK_u)$ algorithm to answer W's keyword ciphertext I_W.

- Keyword Trapdoor Query: A_1 sends a keyword W' to B. B runs the $TrapGen(\Omega, SK_u, W')$ algorithm to answer W''s trapdoor $T_{W'}$.
- Test Query: A_1 selects and sends the keyword ciphertext I_W and trapdoor $T_{W'}$ to B. B executes the $Test(\Omega, T_{W'}, I_W)$ algorithm to return the test result of whether the keyword ciphertext and the trapdoor match.

Challenge: A_1 submits a tuple $(W_0, W_1, {ID_u}^*, {PK_u}^*)$ to B, where W_0 and W_1 are challenging keywords not asked in the previous trapdoor and keyword ciphertext query. If $ID_u^* \neq ID_u^{\diamondsuit}$, B aborts. Otherwise, B picks $\xi \in \{0,1\}$, randomly computes the keyword ciphertext I_{W_ξ}, and returns the challenge ciphertexts I_{W_ξ} to A_1.

Phase 2. A_1 can perform many queries like Phase 1, but A_1 cannot query the keyword ciphertext and trapdoor of W_0 and W_1.

Guess: A_1 outputs $\xi' \in \{0,1\}$. A_1 wins if $\xi' = \xi$. Otherwise, it fails.

Next, the advantages of A_1 in Game 1 are given as

$$Adv_{CKCA-CIND_{A_1}} |\Pr[\xi' = \xi| - \frac{1}{2}|.$$

Game 2.

A_2 simulates the malicious server, and B is a challenger. B and A_2 play this game together. Setup: It only differs from the setup of Game 1 in the following steps. B sends the public keys PK_{O_i}, P_{pub}, and msk to A_2, and SK_{O_i} is unknown to A_2.

Phase 1. The steps are the same as in Phase 1 of Game 1, except for the Secret-Value query and Partial-Private-Key query. The changes in them are as follows:

- Secret-Value query: When A_2 enters the ID_j, if $ID_j = ID_u^{\diamondsuit}$ ($ID_u^{\diamondsuit}$ is the challenge identity), B fails and stops. Otherwise, B returns the secret value corresponding to ID_j.
- Partial-Private-Key-Extract query: When A_2 inputs the identity ID_j, B returns the partial private key corresponding to the ID_j. If PK_j is replaced, B refuses to answer.

Phase 2. Same as Phase 2 of Game 1.

Next, the advantages of A_2 in Game 2 are given as

$$Adv_{CKCA-CIND_{A_2}} |\Pr[\xi' = \xi| - \frac{1}{2}|.$$

Definition 2 (Security of CKCA-CIND). *If the probability that any adversary will win the above two games in polynomial time is negligible, then we state that the CLVPFC-PEKS scheme is CKCA-CIND safe.*

4.3.2. Safety of Trapdoor

The attack methods on the trapdoor include offline keyword-guessing attacks, online keyword-guessing attacks, and insider keyword-guessing attacks.

We first discuss how to defend against offline keyword-guessing attacks. References [46,52,53] have made efforts to resist offline keyword-guessing attacks. They modified the structure of the trapdoor and claimed that if the attacker did not know the server's private key or the receiver's private key, then the scheme would be resistant to offline keyword-guessing attacks by external attackers. In [53], the concept of trapdoor indistinguishability under a choose keyword attack (CKA-TIND) is proposed, and it is proven that CKA-TIND is a sufficient condition to prevent offline keyword guessing attacks. At the same time, ref. [53] proposes a dPEKS scheme with trapdoor indivisibility. The above schemes are proven safe in the random oracle model, but the proof that the scheme is safe in the random oracle model does not necessarily mean that the scheme is safe in reality. Fang et al. [54] proposed a new SCF-PEKS scheme that has no random prediction and asserts that the scheme can safely resist offline keyword guessing attacks by external attackers. However, Lu et al. [19] pointed out that Fang's scheme is insecure under the keyword guessing attack of external attackers.

In the following, we define the concept of CKA-TIND of CLVPFC-PEKS.

Game 3.

A_1 simulates malicious users, and B is the challenger. B and A_1 play this game together.

Setup: A_1 B runs the SetUp(1^k) program to obtain Ω,P_{pub} and msk. B sends P_{pub} to A_1 and keeps msk secret. Then, B sets the key pair of the date user and CSS, i.e., (PK_u, SK_u) and (PK_c, SK_c). Challenger B sends the public key PK_u and PK_c to A_1, while SK_u and SK_c are unknown to A_1.

Phase 1. A_1 executes the User-Public-Key query before executing other queries. The setup lists store the above queries and answers. All lists are initially empty. A_1 makes the queries to challenger B as follows:

- User-Public-Key query: When A_1 inputs the identity ID_i, B outputs the user's public key PK_i .
- Replace-Public-Key query: Same as in Game 1.
- Secret-Value query: Same as in Game 1.
- Partial-Private-Key-Extract query: When A_1 enters the ID_j, if $ID_j = ID_i^{\diamondsuit}$($ID_i^{\diamondsuit}$ is the challenge identity), B fails and stops. Otherwise, B returns the corresponding Partial-Private-Key.
- Keyword Ciphertext Query: Same as in Game 1.
- Keyword Trapdoor Query: Same as in Game 1.
- Test Query: Same as in Game 1.

Challenge: A_1 submits a tuple $(W_0, W_1, \{ID_i{}^*\}, \{PK_i{}^*\})$ $(i \in \{1, 2, \cdots d\})$ to B, where W_0 and W_1 are challenging keywords that were not asked in the previous trapdoor and keyword ciphertext query. If $ID_i^{\diamondsuit} \notin \{ID_i{}^*\}(i \in \{1, 2, \cdots d\})$, B aborts. Otherwise, B picks $\xi \in \{0, 1\}$, randomly computes keyword trapdoor T_{W_ξ}, and returns the challenge trapdoor T_{W_ξ} to the adversary A_1.

Phase 2. A_1 can perform many queries like Phase 1, but A_1 cannot query the keyword ciphertext or trapdoor of W_0 and W_1.

Guess: A_1 outputs $\xi' \in \{0, 1\}$. Adversary A_1 wins if $\xi' = \xi$. Otherwise, it fails.

Next, the advantages of A_1 in Game 3 are given as

$$Adv_{CKA-TIND_{A_1}} |\Pr[\xi' = \xi| - \frac{1}{2}|$$

Game 4.

A_2 simulates the malicious server, and B is a challenger. B and A_2 play this game together.

Setup: This differs from the setup of Game 3 only in the following steps. B sends PK_u, P_{pub}, and msk to A_2, and SK_u is unknown to A_2.

Phase 1. The steps are the same as in Phase 1 of Game 3, except for the Secret-Value and Partial-Private-Key-Extract queries. The changes in them are as follows:

- Secret-Value query: When A_2 enters the ID_j, if $ID_j = ID_i^{\diamondsuit}$ ($ID_i^{\diamondsuit}$ is the challenge identity), B fails and stops. Otherwise, B returns the secret value corresponding to ID_j.
- Partial-Private-Key-Extract query: Same as in Game 2.

Phase 2. Same as Phase 2 of Game 3.

Next, the advantages of A_2 in Game 4 are given as

$$Adv_{CKA-TIND_{A_2}} |\Pr[\xi' = \xi| - \frac{1}{2}|$$

Definition 3 (Security of CKA-TIND)**.** *If the probability that any adversary will win the above two games in polynomial time is negligible, then we state that the CLVPFC-PEKS scheme is CKA-TIND safe.*

Next, we will discuss how to resist online/insider keyword-guessing attacks. Lu et al. [19] pointed out that the main reason for vulnerability to online keyword-guessing attacks is that any opponent can generate legitimate ciphertext for keywords.

Shao et al. [47] pointed out that SCF-PEKS inherently suffers from insider keyword-guessing attacks because malicious servers can run keyword encryption algorithms and test algorithms at the same time. To resist the above two attacks, Wu et al. [55] proposed a searchable public key encryption (SPE-PP) scheme with a privacy protection function. In their plan, the Diffie–Hellman shared secret key is required for the generation of the keyword ciphertext and trapdoor. Specifically, the sender uses the shared key to calculate each keyword ciphertext, while the receiver uses the shared key to generate a trapdoor. An internal adversary (such as a malicious cloud server) cannot obtain the Diffie–Hellman shared secret key, so they cannot construct legal ciphertext to match the trapdoors sent by retrievers for testing. Then, internal adversaries will not be able to implement insider keyword-guessing attacks. On the other hand, only the retriever can use the Diffie–Hellman shared secret key to generate a legitimate trapdoor for each keyword, and the adversary of online keyword guessing cannot obtain the Diffie–Hellman shared secret key. Then, they cannot construct a legal trapdoor to upload to the cloud server. Even if they monitor all the files obtained by the retriever under the public channel, they cannot obtain any keyword-related information through comparison. In conclusion, the reason why the searchable encryption scheme can resist online and insider keyword-guessing attacks is that the keyword ciphertext is embedded in the shared key generated by the sender's private key and the receiver's public key.

In this section, we conduct a comprehensive analysis of the security of the scheme, focusing primarily on the security of the trapdoor and keyword ciphertext. The security of the keyword ciphertext only needs to demonstrate that the scheme is resistant to chosen keyword ciphertext attacks. Trapdoor security is categorized into three aspects: security against offline keyword-guessing attacks, security against online keyword-guessing attacks, and security against internal keyword-guessing attacks. Because CKCA-CIND is a sufficient condition to prevent offline keyword guessing attacks, as long as the scheme is trapdoor-indistinguishable under the chosen keyword attack, then the scheme is secure against the offline keyword guessing attack. Also, as long as the shared key embedded in the keyword ciphertext is generated from the sender's private key and the receiver's public key, the PESK scheme is resistant to both online keyword attacks and internal keyword guessing attacks.

5. The Proposed CLVPFC-PEKS

This system uses a traditional public key encryption algorithm for keywords. However, we will not discuss them in detail here. Therefore, the following algorithms focus mainly on indexing and signature.

SetUp (1^k): Given a security parameter k, this deterministic algorithm outputs the global public parameters Ω and KGC's master secret key (msk). Given k, the KGC performs as follows:

- Choose a k bit prime number q and determine the tuple $\{F_q, E/F_q, G_q, P\}$, where the point P is the generator of G_q.
- Choose a number $x \in_R Z_q^*$ and compute the system public key $P_{pub} = xP$. Set $msk = \{x\}$. Let $(PK, SK) = (P_{pub}, x)$.
- Select five hash functions H_0,H_1,H_2 :$\{0,1\}^* \times G \times G \rightarrow Z_q^*$, $h_1 : \{0,1\}^* \rightarrow Z_q^*$, $h_2 : G_q \rightarrow Z_q^*$.
- Let $\Omega = \{F_q, E/F_q, G_q, P, H_0, H_1, H_2, h_1, h_2, P_{pub}\}$.

KeyGen (Ω, O, U, CSS): Let each EMR have a fixed number of data owners $O = \{O_1, O_2, \cdots, O_d\}$. KGC generates the public/secret key pairs for the CSS, data owner O_i, and retriever U.

- Set-Secret-Value: The participant with $ID_i (i = 1, 2, \cdots, d, C, u)$ selects an element $x_i \in_R Z_q^* (i = 1, 2, \cdots, d, C, u)$ and generates the corresponding public key $P_i = x_i P$ $(i = 1, 2, \cdots, d, C, u)$.

- Extract-Partial-Private-Key: To obtain the partial private key, the user ID_i sends (ID_i, P_i) to the KGC, and then the KGC executes the extraction as in the following steps.
  - Taking the participant's $ID_i (i = 1, 2, \cdots, d, C, u)$ as input, a random number $r_i \in Z_q^* (i = 1, 2, \cdots, d, C, u)$ is selected by KGC and calculates $R_i = r_i P$ $(i = 1, 2, \cdots, d, C, u)$.
  - The KGC computes $e_i = r_i + xH_0(ID_i, R_i, P_i) mod q (i = 1, 2, \cdots, d, C, u)$. The partial private key of the participant with $ID_i (i = 1, 2, \cdots, d, C, u)$ is e_i. The participant with $ID_i (i = 1, 2, \cdots, d, C, u)$ can verify their partial private key by checking whether the equation $Q_i = e_i P = R_i + l_i P_{pub}$ holds, where $l_i = H_0(ID_i, R_i, P_i)$. If the above equation is true, then the private key ID_i is accepted.
- Set-Private-Key: The partial private key of the participant with $ID_i (i = 1, 2, \cdots, d, C, u)$ takes the pair $SK_i = (x_i, e_i)$ as their full private key.
- Set-Public-Key: The participant with $ID_i (i = 1, 2, \cdots, d, C, u)$ takes $PK_i = (P_i, R_i)$ as their full public key.

Enc$(\Omega, F, W, \{ID_i\}, ID, \{SK_{O_i}\}, \{PK_{O_i}\}, PK_u)$:

Step 1: Given the EMR set $F = \{f_1, f_2, \cdots, f_n\}$ with corresponding identities $ID = \{id_1, id_2, \cdots, id_n\}$, it will be encrypted as the ciphertext set $C = \{c_1, c_2, \cdots, c_n\}$ through the traditional public key encryption algorithm. To generate the multi-signature on the encrypted file $c_t \in C (1 \leqslant t \leqslant n)$, each signer $O_i (1 \leqslant i \leqslant d)$ does the following:

- O_i chooses a number $y_{i,t} \in_R Z_q^*$ and computes $Y_{i,t} = y_{i,t} P$.
- O_i broadcasts $Y_{i,t}$ to other members $O_k (1 \leqslant k \leqslant d, k \neq i)$ of the group.
- Computes $Y_t = \sum_{i=1}^{d} Y_{i,t}$, $P_0 = \sum_{i=1}^{d} P_i$, $Q_0 = \sum_{i=1}^{d} Q_i$.
- Computes $h_t = H_1(c_t, id_t, Y_t, P_0)$ and $h_t' = H_1(c_t, id_t, Y_t, Q_0)$.
- O_i computes $V_{i,t} = y_{i,t} + h_t x_i + h_t' e_i$, generates a signature $sig_{i,t} = (Y_{i,t}, V_{i,t})$ for c_t, and then sends $V_{i,t}$ to the designated clerk O_z.
- Upon receiving $V_{i,t}$, O_z computes $V_t = \sum_{i=1}^{d} V_{i,t}$ and outputs the signature $sig_t = \{Y_t, V_t\}$. Let $sig = \{sig_1, \cdots, sig_n\}$, where $V_t P = Y_t + h_t P_0 + h_t' Q_0$.

Step 2: All users specify a user to generate an index, for example, O_d. O_d runs this algorithm to generate the index of file set F. Given the keyword set W, O_d builds an index for each file $f_i \in F$. The index for each f_i is generated based on the keyword field $W = \{w_{i1}, w_{i2}, \cdots, w_{im}\}$, where m is a fixed integer. O_d randomly selects $\xi \in Z_q^*$ and calculates $\xi + x_d$ to O_1 through public key encryption. O_1 computes $\xi + x_d + x_1$ and sends it to O_2 through public key encryption. O_2 computes $\xi + x_d + x_1 + x_2$ and sends it to O_3 through public key encryption, and so on until O_{d-1} computes $\xi + x_d + x_1 + \cdots + x_{d-1}$ and sends it to O_d through public key encryption. O_d calculates $x_0 = \xi + x_d + x_1 + \cdots + x_{d-1} - \xi = x_1 + x_2 + \cdots + x_d$, and calculates $e_0 = e_1 + e_2 + \cdots + e_d$ in the same way as O_j. Let $R_0 = \sum_{t=1}^{d} R_t$, $l_0 = \sum_{t=1}^{d} l_t$. Construct an m-degree polynomial with the following equation:

$$F(x) = b_{i,m} x^m + b_{i,m-1} x^{m-1} + \cdots + b_{i,1} x + b_{i,0},$$

so $h_2(t)h_1(w_{i,1}), h_2(t)h_1(w_{i,2}), \cdots, h_2(t)h_1(w_{i,m})$ is the root of equation $F(x) = 1$, where $t = (x_0 + e_0)P_u + x_0 R_u + x_0 l_u P_{pub}$. Then, O_j selects $\lambda_i, \mu_i \in_R Z_q{}^*$ and computes $M_i = \lambda_i Q_c$, set $I_{i,1} = M_i - \mu_i P$, $I_{i,2} = \lambda_i P$, $V_{i,j} = \mu_i b_{i,j} (0 \leqslant j \leqslant m)$, and the index set is $I = \{I_1, \cdots, I_n\}$, where $I_i = \{I_{i,1}, I_{i,2}, V_{i,0}, V_{i,1}, \cdots, V_{i,m}\}$. Finally, O_d sends I to CSS.

TrapGen(Ω, SK_u, W'): The DO calculates the value of P_0, R_0, l_0 as follows: $P_0 = \sum_{i=1}^{d} P_i$, $R_0 = \sum_{i=1}^{d} R_i$, $l_0 = \sum_{i=1}^{d} l_i$. Given the queried keywords set $W' = \{w_1', w_2', \cdots, w_l'\}$, the DO U first select an element $\eta \in_R Z_q^*$ and set $T_{W'_{m+1}} = \eta P$,

$T_{W'_j} = l^{-1}h_2(t)^j \sum_{r=1}^{l} h_1(w'_r)^j P + \eta P_C$, where $0 \leqslant j \leqslant m$,$t = (x_u + e_u)P_0 + x_u R_0 + x_u l_0 P_{pub}$. Finally, they send $T_{w'} = \{T_{w'_0}, T_{w'_1}, \cdots, T_{w'_m}, T_{w'_{m+1}}\}$ to the CSS.

Test$(\Omega, T_{W'}, I, C)$: After gaining the search token $T_{W'}$, the CSS computes $M_i = \lambda_i Q_C$ fist, and then verifies whether Equation (1) holds.

$$I_{i,1} + \sum_{j=0}^{m} V_{i,j}(T_{w'_j} - x_C T_{w'_{m+1}}) = M_i \tag{1}$$

If Equation (1) holds, the CSS will return the relevant ciphertext set $C' = \{c_{k_1}, c_{k_2}, \cdots, c_{k_s}\}$ and its corresponding identity set $ID' = \{id_{k_1}, id_{k_2}, \cdots, id_{k_s}\}$ to PAS. Otherwise, it returns $\perp$. The specific test process is shown in Algorithm 1.

Algorithm 1: Search over encrypted data

Input: Trapdoor $T_{W'}$, index I, ciphertext C, secret key SK_C, and public parameters Ω.
Output: Search results C' and corresponding identity set ID' or $\perp$.

- $T_{w'} = \{T_{w'_0}, T_{w'_1}, \cdots, T_{w'_m}, T_{w'_{m+1}}\}$
- $I = \{I_1, I_2, \cdots, I_n\}, I_i = \{I_{i,1}, I_{i,2}, V_{i,0}, V_{i,1}, \cdots, V_{i,m}\}$.
- $M_i = e_C I_{i,2}$
- $for\ 0 \leqslant i \leqslant n\ do\ I_{i,1} + \sum_{j=0}^{m} V_{i,j}(T_{w'_j} - x_C T_{w'_{m+1}}) = M_i$
- If Equation (1) holds, CSS returns the ciphertext c_{k_i}; otherwise, it returns $\perp$;
- end for
- CSS returns the relevant results C' and corresponding identity set ID' or $\perp$ to PAS.

Verify $(\Omega, C', ID', sig, \{PK_{O_i}\})$: After receiving the search results C' , PAS computes the proof information (ϕ_1, ϕ_2) and σ. Finally, PAS verifies whether Equation (2) holds. The specific verify process is shown in Algorithm 2.

Algorithm 2: Results verification.

Input: Search results C' with corresponding identity set ID', public key set $\{PK_{O_i}\}$, signature $sig = \{sig_1, \cdots, sig_n\}$, and public parameters Ω, where $sig_t = \{Y_t, V_t\}$.
Output: "Accept" or "Reject"

- $C' = \{c_{k_1}, c_{k_2}, \cdots, c_{k_s}\}$, $ID' = \{d_{k_1}, id_{k_2}, \cdots, id_{k_s}\}$;
- public key set $\{PK_{O_1}, PK_{O_2}, \cdots, PK_{O_d}\}$;
- $sig = \{sig_1, \cdots, sig_n\}$, $sig_t = \{Y_t, V_t\}$;
- compute $P_0 = \sum_{t=1}^{d} P_t$, $Q_0 = \sum_{t=1}^{d} Q_t$;
- compute $\phi_1 = \sum_{\tau=1}^{s} H_1(c_{k_\tau}, id_{k_\tau}, Y_{k_\tau}, P_0)$, $\phi_2 = \sum_{\tau=1}^{s} H_2(c_{k_\tau}, id_{k_\tau}, Y_{k_\tau}, Q_0)$, $\sigma = \sum_{\tau=1}^{s} V_{k_\tau}$;
- Check
$$\sigma P = \sum_{\tau=1}^{s} Y_{k_\tau} + \phi_1 P_0 + \phi_2 Q_0; \tag{2}$$
- If Equation (2) holds, output "Accept" and send C' to retriever; otherwise, output "Reject".

6. Security of Scheme

We will analyze the correctness and security of the CLVPFC-PEKS scheme in this section.

6.1. Correctness

Theorem 1. *The CLVPFC-PEKS scheme is computationally consistent.*

Proof. For the correctness of our CLVPFC-PEKS scheme, we do two things. First, we illustrate that the CSS can effectively ascertain a match between the keyword ciphertext's index and the trapdoor when the keyword set $W' \subseteq W$, where W' is a set of keywords searched by a specific user and W is the keyword set of ciphertext. Subsequently, we elucidate that, given that the search outcomes successfully traverse the established result verification protocol, retrievers can ascertain the accuracy of the search results.

During the test phase, the CSS obtains the index $I = \{I_1, I_2, \cdots, I_n\}$ and trapdoor $T_{w'} = \{T_{w'_0}, T_{w'_1}, \cdots, T_{w'_m}, T_{w'_{m+1}}\}$. The CSS starts performing computations.

$$
\begin{aligned}
& e_C I_{i,2} = (r_C + x l_C)\lambda_{\mathrm{i}} P = \lambda_{\mathrm{i}}\, Q_C = M_i\,. \\
& I_{i,1} + \sum_{j=0}^{m} V_{i,j}(T_{w'_j} - x_C T_{w'_{m+1}}) \\
&= M_i - \mu_i P + \sum_{j=0}^{m} \mu_i b_{ij}[l^{-1}h_2(t)^j \sum_{r=1}^{l} h_1(w'_r)^j P + \eta P_C - x_C \eta P] \\
&= M_i - \mu_i P + \sum_{j=0}^{m} \mu_i b_{ij}(l^{-1}h_2(t)^j \sum_{r=1}^{l} h_1(w'_r) P) \\
&= M_i - \mu_i P + l^{-1}\mu_i \sum_{j=0}^{m} b_{ij} h_2(t)^j \sum_{r=1}^{l} h_1(w'_r)^j P \\
&= M_i - \mu_i P + l^{-1}\mu_i [\sum_{j=0}^{m} b_{ij} h_2(t)^j h_1(w'_1)^j + \cdots + \sum_{j=0}^{m} b_{ij} h_2(t)^j h_1(w'_l)^j] P
\end{aligned}
$$

If $W' \subseteq W$, then $h_2(t)h_1(w'_1), \cdots, h_2(t)h_1(w'_l)$ are the root of the equation $F(x) = 1$, where $F(x) = b_{i,m}x^m + b_{i,m-1}x^{m-1} + \cdots + b_{i,1}x + b_{i,0}$. Thus,

$$
\begin{aligned}
& I_{i,1} + \sum_{j=0}^{m} V_{i,j}(T_{w'_j} - x_C T_{w'_{m+1}}) \\
&= M_i - \mu_i P + l^{-1}\mu_i [\sum_{j=0}^{m} b_{ij} h_2(t)^j h_1(w'_1)^j + \cdots + \sum_{j=0}^{m} b_{ij} h_2(t)^j h_1(w'_l)^j] P \\
&= M_i - \mu_i P + l^{-1}\mu_i (1 + 1 + \cdots + 1) P \\
&= M_i - \mu_i P + \mu_i P \\
&= M_i
\end{aligned}
$$

Equation (1) is satisfied, and thereby CSS can correctly test whether the keyword ciphertext matches the trapdoor.

In the test phase, the PAS obtains signature $sig = \{sig_1, sig_2, \cdots, sig_n\}$ and ciphertext $C' = \{\mathrm{c}_{k_1}, \mathrm{c}_{k_2}, \cdots, \mathrm{c}_{k_s}\}$, computing

$$\phi_1 = \sum_{\tau=1}^{s} H_1(c_{k_\tau}, id_{k_\tau}, Y_{k_\tau}, P_0), \phi_2 = \sum_{\tau=1}^{s} H_2(c_{k_\tau}, id_{k_\tau}, Y_{k_\tau}, Q_0)$$

obtaining the proof information (ϕ_1, ϕ_2), and then continuing to calculate

$$
\begin{aligned}
\sigma P &= \sum_{\tau=1}^{s} V_{k_\tau} P = \sum_{\tau=1}^{s} (Y_{k_\tau} + h_{k_\tau} P_0 + h'_{k_\tau} Q_0) \\
&= \sum_{\tau=1}^{s} Y_{k_\tau} + \sum_{\tau=1}^{s} h_{k_\tau} P_0 + \sum_{\tau=1}^{s} h'_{k_\tau} Q_0 \\
&= \sum_{\tau=1}^{s} Y_{k_\tau} + \sum_{\tau=1}^{s} H_1(c_{k_\tau}, id_{k_\tau}, Y_{k_\tau}, P_0) P_0 + \sum_{\tau=1}^{s} H_1(c_{k_\tau}, id_{k_\tau}, Y_{k_\tau}, Q_0) Q_0.
\end{aligned}
$$

If $C' \subseteq C$, then

$$\phi_1 = \sum_{\tau=1}^{s} H_1(c_{k_\tau}, id_{k_\tau}, Y_{k_\tau}, P_0) = \sum_{\tau=1}^{s} H_1(c_{\rho(\tau)}, id_{\rho(\tau)}, Y_{\rho(\tau)}, P_0)$$

$$\phi_2 = \sum_{\tau=1}^{s} H_1(c_{k_\tau}, id_{k_\tau}, Y_{k_\tau}, Q_0) = \sum_{\tau=1}^{s} H_1(c_{\rho(\tau)}, id_{\rho(\tau)}, Y_{\rho(\tau)}, Q_0)$$

where $\rho(\tau) \in [1, n]$. So we have $\sigma P = \sum_{\tau=1}^{s} Y_{k_\tau} + \phi_1 P_0 + \phi_2 Q_0$. Equation (2) in program 2 holds. We can also make sure that the ciphertext cannot be modified. □

6.2. Security

Lemma 1. *Assuming that adversary A_1 can triumph in Game 1, then it is feasible to devise algorithm B, aimed at resolving the ECDDDH problem.*

Proof. Let us hypothesize that the tuple (P, aP, bP, X) constitutes an instance of the ECDDDH problem. In order to ascertain whether $X = abP$, algorithm B will assume the role of the challenger.

Setup: B executes the setup (1^k) procedure to obtain public parameters $\Omega = \{F_q, E/F_q, G_q, P, G, H_0, H_1, H_2, P_{pub}\}$ along with $msk = \{x\}$ and $P_{pub} = xP$. B then forwards the parameter Ω to A_1 while keeping msk secret. B selects $x_i \in Z_q^*(i \in \{2, \cdots, d, C\})$, $r_i \in Z_q^*(i \in \{1, 2, \cdots, d, C\})$ randomly and sets

$P_i = x_i P$ $(i \in \{2, \cdots, d, C\})$, $P_1 = aP$,$R_i = r_i P$ $(i \in \{1, 2, \cdots, d, C\})$, and $e_i = r_i + xH_0(ID_i, R_i, P_i) \bmod q(i \in \{1, 2, \cdots, d, C\})$.

B sends Ω, PK_{O_i} $(i \in \{1, 2, \cdots, d\})$, and PK_C to A_1, but SK_{O_i} $(i \in \{1, 2, \cdots, d\})$ and SK_C are unknown to A_1.

Phase 1: Prior to conducting other queries, execute the user's public key query utilizing the identity ID_u. Set up multiple lists to record the queries and corresponding responses. Each list starts off empty.

For a user public key query, B keeps a list L_u of the tuple $(ID_u, x_u P, r_u P, r_u)$ and, upon receiving an identity ID_u, performs the following steps.

Case 1. $ID_u = ID_u^{\diamondsuit}$. $x_u^{\diamondsuit} \in Z_q^*$, setting $PK_u^{\diamondsuit} = (x_u^{\diamondsuit} P, bP)$, and adding the tuple $(ID_u^{\diamondsuit}, x_u^{\diamondsuit} P, x_u^{\diamondsuit}, bP, \diamondsuit)$ to the list L_u, where $\diamondsuit$ represents a null value.

Case 2. $ID_u \neq ID_u^{\diamondsuit}$. B randomly chooses two different numbers $x_u, r_u \in Z_q^*$, setting $PK_u = (x_u P, r_u P)$, and adds the tuple $(ID_u, x_u P, r_u P, r_u)$ to the list L_u.

Replace-Public-Key query: B keeps a list L_R of tuple $(ID_j, PK_j, PK_j{}')$. When A_1 inputs $(ID_j, PK_j{}')$, B replaces PK_j with $PK_j{}'$ and adds $(ID_j, PK_j, PK_j{}')$ to the list L_R.

Secret-Value query: When A_1 receives the request for the secret value associated with ID_j, B finds $(ID_j, x_j P, r_j P, r_j)$ in the list L_u and returns x_j. If PK_j is replaced, B refuses to answer.

Partial-Private-Key query: B establishes a list L_e of tuple (ID_j, e_j) when A_1 asks for the partial private key of ID_j. If $ID_j = ID_u^{\diamondsuit}$, B fails and stops. Otherwise, B finds $(ID_j, x_j P, r_j P, r_j)$ in the list L_u, and runs the Extract-Partial-Private-Key algorithm, generating e_j. B outputs e_j and adds (ID_j, e_j) to the list L_e.

Keyword Ciphertext Query: When A_1 asks $W = \{w_{i,1}, w_{i,2}, \cdots, w_{i,m}\}$ for the keyword ciphertext, B operates the $Enc(\Omega, F, W, \{ID_i\}, ID, \{SK_{O_i}\}, \{PK_{O_i}\}, PK_u)$ algorithm to generate keyword ciphertext $I_W = \{I_{i,1}, I_{i,2}, V_{i,0}, V_{i,1}, \cdots, V_{i,m}\}$.

Keyword Trapdoor Query: When A_1 asks $W' = \{w_1', w_2', \cdots, w_l'\}$ for the trapdoor, B operates the $TrapGen\{\Omega, SK_u, W'\}$ algorithm to generate the trapdoor $T_{w'} = \{T_{w'_0}, T_{w'_1}, \cdots, T_{w'_m}, T_{w'_{m+1}}\}$.

Test Query: A_1 gives the keyword ciphertext I_W and keyword trapdoor $T_{w'}$, and B compares them using Algorithm 1.

Challenge: A_1 submits a tuple $(W_0, W_1, ID_u{}^*, PK_u^*)$ to B, where $W_0 = \{w_{0,1}, w_{0,2}, \cdots, w_{0,m}\}$ and $W_1 = \{w_{1,1}, w_{1,2}, \cdots, w_{1,m}\}$ are challenging keywords not requested in the previous trapdoor and keyword ciphertext query. If $ID_u^* \neq ID_u^{\diamondsuit}$, B aborts. Otherwise, $ID_u^* = ID_u^{\diamondsuit}$, B calculates $l_u{}^{\diamondsuit} = H_0(ID_u^{\diamondsuit}, R_u^{\diamondsuit}, P_u^{\diamondsuit})$ and picks $\xi \in \{0, 1\}$ randomly. B computes

$$t = (\sum_{k=2}^{d} x_k + \sum_{k=1}^{d} e_k) P_u^{\diamondsuit} + \sum_{k=2}^{d} x_k R_u^{\diamondsuit} + \sum_{k=2}^{d} x_k l_u^{\diamondsuit} P_{pub} + x_u^{\diamondsuit} aP + l_u^{\diamondsuit} xaP + X.$$

Let $F(x) = (x - h_2(t)h_1(w_{\xi,1}))(x - h_2(t)h_1(w_{\xi,2})) \cdots (x - h_2(t)h_1(w_{\xi,m})) - 1$, which can obtain $F(x) = b_{\xi,m}x^m + b_{\xi,m-1}x^{m-1} + \cdots + b_{\xi,1}x + b_{\xi,0}$ by combining similar terms. Then, B selects $\lambda_\xi, \mu_\xi \in_R {Z_q}^*$ and computes $M_\xi = \lambda_\xi Q_c$. Set $I_{\xi,1} = M_\xi - \mu_\xi P$, $I_{\xi,2} = \lambda_\xi P$, $V_{\xi,j} = \mu_\xi b_{\xi,j} (0 \leqslant j \leqslant m)$. Thus, the corresponding keyword ciphertext of $W_\xi = \{w_{\xi,1}, w_{\xi,2}, \cdots, w_{\xi,m}\}$ is $I_{W\xi} = \{I_{\xi,1}, I_{\xi,2}, V_{\xi,0}, V_{\xi,1}, \cdots, V_{\xi,m}\}$. B returns the challenge ciphertexts I_{W_ξ} to the adversary A_1.

Phase 2: A_1 can continue to execute various queries, but there is a limitation that A_1 is not allowed to query the keyword ciphertext or trapdoor of W_0 or W_1.

Guess: A_1 returns ξ'.

Solve CDH problem: If $\xi' = \xi$, B returns 1, otherwise 0. If $X = abP$, then

$$t = (\sum_{k=2}^{d} x_k + \sum_{k=1}^{d} e_k)P_u^\diamondsuit + \sum_{k=2}^{d} x_k R_u^\diamondsuit + \sum_{k=2}^{d} x_k l_u^\diamondsuit P_{pub} + x_u^\diamondsuit aP + l_u^\diamondsuit xaP + X$$
$$= (x_0 + e_0)P_u^\diamondsuit + \sum_{k=2}^{d} x_k R_u^\diamondsuit + x_0 l_u^\diamondsuit P_{pub} + abP$$
$$= (x_0 + e_0)P_u^\diamondsuit + x_0 R_u^\diamondsuit + x_0 l_u^\diamondsuit P_{pub}$$

Therefore, $I_{W\xi}$ is a valid keyword ciphertext. Suppose that the advantage of A_1 winning in the above game is ε. So,

$$\Pr[\xi' = \xi | X = abP] = \frac{1}{2} + \varepsilon.$$

If $X \neq abP$, then I_{W_ξ} is an invalid keyword ciphertext. A_1 has no advantage in distinguishing $\xi = 0$ from $\xi = 1$. Hence,

$$\Pr[\xi' = \xi | X = abP] = \frac{1}{2}.$$

Probability: Let q_u, q_r and q_e be the number of the user public key query, Replace-Public-Key query, and the Partial-Private-Key query, respectively. The two events are as follows:

π_1 : A_1 did not replace $ID_u^\diamondsuit$'s public key $R_u^\diamondsuit$ and queried the partial-private-key for $ID_u^\diamondsuit$.

$\pi_2 : ID_u^* = ID_u^\diamondsuit$.

It is not hard to obtain the following results.

$\Pr[\pi_1] = \frac{q_u - q_r - q_e}{q_u}$,

$\Pr[\pi_2 | \pi_1] = \frac{1}{q_u - q_r - q_e}$, $\Pr[B\ success] = \Pr[\pi_1 \wedge \pi_2] = \frac{1}{q_u}$.

If A_1 wins Game 1 with an advantage of ε, then B has a probability greater than $\frac{\varepsilon}{q_u}$ to determine whether $X = abP$. □

Lemma 2. *Assuming that adversary A_2 can win Game 2, algorithm B can constructed to solve the ECDDDH problem by exploiting the adversary's ability.*

Proof. Suppose that the tuple (P, aP, bP, X) is an example of an ECDDDH problem. To determine whether $X = abP$, B will play the part of the challenger.

Setup: B runs the setup (1^k) program to obtain public parameters Ω $msk = \{x\}$ and $P_{pub} = xP$. B selects $x_i \in Z_q^* (i \in \{1, 2, \cdots, d, C\})$, $r_i \in Z_q^* (i \in \{2, \cdots, d, C\})$ randomly, and sets

$$P_i = x_i P \ (i \in \{1, 2, \cdots, d, C\}), \qquad R_1 = aP, R_i = r_i P \ (i \in \{2, \cdots, d, C\}),$$
$$e_1 = a + xH_0(ID_1, R_1, P_1) \bmod q, \qquad e_i = r_i + xH_0(ID_i, R_i, P_i) \bmod q (i \in \{2, \cdots, d, C\})$$

B sends Ω, $PK_{O_i} (i \in \{1, 2, \cdots, d\})$, and (PK, SK) to A_2, while $SK_{O_i} (i \in \{1, 2, \cdots, d\})$ are unknown to A_2.

Phase 1: Execute the user's public key query before other queries using the identity ID_u. Set up multiple lists to store the queries and answers. All lists are initially empty.

User public key query: B maintains a list L_u containing the tuple $(ID_u, x_u P, r_u P, r_u)$ and takes the following actions when receiving an identity ID_u:

Case 1. $ID_u = ID_u^\diamondsuit$. B chooses a number $r_u^\diamondsuit \in Z_q^*$ at random, sets $PK_u^\diamondsuit = (bP, r_u^\diamondsuit P)$, and adds the tuple $(ID_u^\diamondsuit, bP, \diamondsuit, r_u^\diamondsuit P, r_u^\diamondsuit)$ to the list L_u, where $\diamondsuit$ represents a null value.

Case 2. $ID_u \neq ID_u^\diamondsuit$. B chooses $x_u, r_u \in Z_q^*$ at random, sets $PK_u = (x_u P, r_u P)$, and adds the tuple $(ID_u, x_u P, r_u P, r_u)$ to the list L_u.

Replace-Public-Key query: Same as in Lemma 1.

Secret-Value query: B established a list L_s of tuple (ID_j, x_j). When A_2 asks the secret value for ID_j. If $ID_j = ID_u^{\diamondsuit}$, B fails and stops. Otherwise, B finds (ID_j, x_jP, r_jP, r_j) in list L_u, and returns x_j.

Partial-Private-Key query: When A_2 asks the partial private key of ID_j, B finds (ID_j, x_jP, r_jP, r_j) in list L_u, running the Extract-Partial-Private-Key algorithm and returning e_j. If PK_j is replaced, B refuses to answer.

Keyword Ciphertext Query: Same as in Lemma 1.

Keyword Trapdoor Query: Same as in Lemma 1.

Test Query: Same as in Lemma 1.

Challenge: A_2 submits a tuple $(W_0, W_1, ID_u^*, PK_u^*)$ that meets the requirements of Game 2, where $W_0 = \{w_{0,1}, w_{0,2}, \cdots, w_{0,m}\}$ and $W_1 = \{w_{1,1}, w_{1,2}, \cdots, w_{1,m}\}$ are challenging keywords not asked in the previous trapdoor query and keyword ciphertext query. If $ID_u^* \neq ID_u^{\diamondsuit}$, B aborts. Otherwise, $ID_u^* = ID_u^{\diamondsuit}$, B computes $l_u^{\diamondsuit} = H_0(ID_u^{\diamondsuit}, R_u^{\diamondsuit}, P_u^{\diamondsuit})$, $l_1 = H_0(ID_1, R_1, P_1)$ and picks $\xi \in \{0, 1\}$ randomly. B computes

$$t = (\sum_{k=1}^{d} x_k + xl_1 + \sum_{k=2}^{d} e_k)P_u^{\diamondsuit} + \sum_{k=1}^{d} x_k R_u^{\diamondsuit} + \sum_{k=1}^{d} x_k l_u^{\diamondsuit} P_{pub} + X.$$

Let $F(x) = (x - h_2(t)h_1(w_{\xi,1}))(x - h_2(t)h_1(w_{\xi,2})) \cdots (x - h_2(t)h_1(w_{\xi,m})) - 1$, which can obtain $F(x) = b_{\xi,m}x^m + b_{\xi,m-1}x^{m-1} + \cdots + b_{\xi,1}x + b_{\xi,0}$ by combining similar terms. Then, select $\lambda_\xi, \mu_\xi \in_R {Z_q}^*$ at random and compute $M_\xi = \lambda_\xi Q_c$. Set $I_{\xi,1} = M_\xi - \mu_\xi P$, $I_{\xi,2} = \lambda_\xi P$, $V_{\xi,j} = \mu_\xi b_{\xi,j} (0 \leqslant j \leqslant m)$, and thus $W_\xi = \{w_{\xi,1}, w_{\xi,2}, \cdots, w_{\xi,m}\}$'s keyword ciphertext is $I_{W_\xi} = \{I_{\xi,1}, I_{\xi,2}, V_{\xi,0}, V_{\xi,1}, \cdots, V_{\xi,m}\}$. B returns the challenge ciphertexts I_{W_ξ} to the adversary A_2.

Phase 2: Attacker A_2 can continue to execute various queries, but there is a limitation that attacker A_2 is not allowed to query the keyword ciphertext or trapdoor of W_0 or W_1.

Guess: A_2 returns ξ'.

Solve the ECDDDH problem. If $\xi' = \xi$, B returns 1. Otherwise, 0. If $X = abP$, then

$$t = (\sum_{k=1}^{d} x_k + xl_1 + \sum_{k=2}^{d} e_k)P_u^{\diamondsuit} + \sum_{k=1}^{d} x_k R_u^{\diamondsuit} + \sum_{k=1}^{d} x_k l_u^{\diamondsuit} P_{pub} + X$$

$$t = (\sum_{k=1}^{d} x_k + \sum_{k=1}^{d} e_k)P_u^{\diamondsuit} + \sum_{k=1}^{d} x_k R_u^{\diamondsuit} + \sum_{k=1}^{d} x_k l_u^{\diamondsuit} P_{pub}$$

$$= (x_0 + e_0)P_u^{\diamondsuit} + x_0 R_u^{\diamondsuit} + x_0 l_u^{\diamondsuit} P_{pub}$$

Therefore, I_{W_ξ} is a valid keyword ciphertext. Suppose that the advantage of A_2 wins in the above game is ε, so

$$\Pr[\xi' = \xi | X = abP] = \frac{1}{2} + \varepsilon.$$

If $X \neq abP$, then I_{W_ξ} is an invalid keyword ciphertext. A_2 has no advantage in distinguishing $\xi = 0$ from $\xi = 1$. Hence,

$$\Pr[\xi' = \xi | X = abP] = \frac{1}{2}.$$

Probability: Let q_u, q_r, q_s be the number of user public key queries, Replace-Public-Key queries, and Secret-Value queries, respectively. The two events are as follows:

π_1: A_2 did not replace $ID_u^{\diamondsuit}$'s public key $P_u^{\diamondsuit}$ nor perform the Secret-Value query for $ID_u^{\diamondsuit}$.

$\pi_2 : ID_u^* = ID_u^{\diamondsuit}$.

It is not hard to obtain the following results.

$\Pr[\pi_1] = \frac{q_u - q_r - q_s}{q_u}$,

$\Pr[\pi_2 | \pi_1] = \frac{1}{q_u - q_r - q_s}$,

$\Pr[B\ success] = \Pr[\pi_1 \wedge \pi_2] = \frac{1}{q_u}$.

If A_2 has an ε advantage to win the game, then B has a probability greater than $\frac{\varepsilon}{q_u}$ to determine whether $X = abP$. □

Theorem 2. *Our CLVPFC-PEKS scheme is CKCA-CIND secure in a standard model if the ECDDDH problem is hard.*

Proof. Theorem 2 holds from Lemma 1 and Lemma 2. □

Lemma 3. *Assuming the adversary A_1 can win Game 3, then algorithm B can be constructed to solve the ECDDDH problem.*

Proof. Suppose that the tuple (P, aP, bP, X) is an example of an ECDDDH problem. To determine whether $X = abP$, B will play the part of the challenger.

Setup: B runs the setup (1^k) program to obtain the public parameters $\Omega = \{F_q, E/F_q, G_q, P, G, H_0, H_1, H_2, P_{pub}\}$, where $msk = \{x\}$ and $P_{pub} = xP$. Then, B randomly selects $r_u, x_C, r_C \in Z_q^*$, and sets

$$P_u = aP, R_u = r_uP \; P_C = x_CP, R_C = r_CP,$$
$$e_C = r_C + xH_0(ID_C, R_C, P_C) \bmod q,$$
$$e_u = r_u + xH_0(ID_u, R_u, P_u) \bmod q\ .$$

B sends Ω, PK_u and PK_C to A_1, but SK_u and SK_C are unknown to A_1.

Phase 1: Execute the user's public key query before other queries using the identity ID_i. Set up multiple lists to store the queries and answers. All lists are initially empty.

User public key query: B keeps a list L_o of the tuple (ID_i, x_iP, r_iP, r_i), and upon receiving an identity ID_i, performs the following steps.

Case 1. $ID_i = ID_i^{\diamondsuit}$, B randomly chooses a number $x_i^{\diamondsuit} \in Z_q^*$, setting $PK_i^{\diamondsuit} = (x_i^{\diamondsuit}P, bP)$, and adds the tuple $(ID_i^{\diamondsuit}, x_i^{\diamondsuit}P, x_i^{\diamondsuit}, bP, \diamondsuit)$ to the list L_o, where $\diamondsuit$ represents a null value.

Case 2. $ID_i \neq ID_i^{\diamondsuit}$, B randomly chooses two different numbers $x_i, r_i \in Z_q^*$, setting $PK_i = (x_iP, r_iP)$, and adds the tuple (ID_i, x_iP, r_iP, r_i) to the list L_o.

Replace-Public-Key query: Same as in Lemma 1.

Secret-Value query: Same as in Lemma 1.

Partial-Private-Key query: B establishes a list L_e of tuple (ID_j, e_j) when A_1 asks for the partial private key of ID_j. If $ID_j = ID_i^{\diamondsuit}$, B fails and stops. Otherwise, B finds (ID_j, x_jP, r_jP, r_j) in the list L_o, and runs the Extract-Partial-Private-Key algorithm, generating e_j. B outputs e_j and adds (ID_j, e_j) to the list L_e.

Keyword Ciphertext Query: Same as in Lemma 1.

Keyword Trapdoor Query: Same as in Lemma 1.

Test Query: Same as in Lemma 1.

Challenge: A_1 submits a tuple $(W_0', W_1', \{ID_1{}^*, \cdots, ID_d{}^*\}, \{PK_1{}^*, \cdots, PK_d{}^*\})$, where $W_0' = \{w_{0,1}', w_{0,2}', \cdots, w_{0,l}'\}$ and $W_1' = \{w_{1,1}', w_{1,2}', \cdots, w_{1,l}'\}$ are challenging keywords that are not requested in the previous trapdoor and keyword ciphertext query. If $ID_i^{\diamondsuit} \notin \{ID_1{}^*, ID_2{}^*, \cdots, ID_d{}^*\}$, B aborts. Otherwise, $ID_i^{\diamondsuit} \in \{ID_1{}^*, ID_2{}^*, \cdots, ID_d{}^*\}$. Without losing generality, it is better to set ID_1^* as $ID_i^{\diamondsuit}$. B calculates $l_0{}^* = \sum_{i=1}^{d} H_0(ID_i^*, R_i^*, P_i^*)$. B picks $\xi \in \{0,1\}$ randomly, and computes

$$t = e_u \sum_{i=1}^{d} P_i{}^* + \sum_{i=2}^{d} x_i{}^* aP + \sum_{i=1}^{d} r_i{}^* aP + l_0{}^* xaP + X$$

B selects an element $\eta_\xi \in_R Z_q^*$ and sets $T_{W'_{\xi,m+1}} = \eta_\xi P$,

$T_{W'_{\xi,j}} = l^{-1}h_2(t)^j \sum_{r=1}^{l} h_1(w'_{\xi,j})^j P + \eta_\xi P_C$, where $0 \leqslant j \leqslant m$. Finally, B sends $T_{W'_\xi} = \{T_{w'_\xi,0}, T_{w'_\xi,1}, \cdots, T_{w'_\xi,m}, T_{w'_{\xi,m+1}}\}$ to the adversary A_1.

Phase 2: Attacker A_1 can continue to execute various queries, but there is a limitation that attacker A_1 is not allowed to query the keyword ciphertext or trapdoor of W_0 or W_1.

Guess: A_1 returns ξ'.

Solve CDH problem: If $\xi' = \xi$, B returns 1, otherwise 0. If $X = abP$, then

$$\begin{aligned} t &= e_u \sum_{i=1}^{d} P_i{}^* + \sum_{i=1}^{d} x_i{}^* aP + \sum_{i=2}^{d} r_i{}^* aP + l_0{}^* xaP + X \\ &= (x_u + e_u)P_0^* + \sum_{i=2}^{d} r_i{}^* aP + x_u l_0^* P_{pub} + abP \\ &= (x_u + e_u)P_0^* + x_u R_0^* + x_u l_0^* P_{pub}. \end{aligned}$$

Therefore, $T_{W'_\xi}$ is a valid keyword ciphertext. Suppose that the advantage of A_1 winning in the above game is ε. So,

$$\Pr[\xi' = \xi | X = abP] = \frac{1}{2} + \varepsilon.$$

If $X \neq abP$, then$T_{W'_\xi}$ is an invalid trapdoor. A_1 has no advantage in distinguishing $\xi = 0$ from $\xi = 1$. Hence,

$$\Pr[\xi' = \xi | X = abP] = \frac{1}{2}.$$

Probability: Let q_o, q_r, and q_e be the number of the User public key queries, Replace-Public-Key queries, and Partial-Private-Key queries, respectively. The two events are as follows:

π_1: A_1 did not replace $ID_i^\diamondsuit$'s public key $R_i^\diamondsuit$ and queries the partial-private-key for $ID_i^\diamondsuit$.

π_2: $ID_1^* = ID_i^\diamondsuit$.

It is not hard to obtain the following results.

$\Pr[\pi_1] = \frac{q_o - q_r - q_e}{q_o}$,

$\Pr[\pi_2 | \pi_1] = \frac{1}{q_o - q_r - q_e}$,

$\Pr[B\ success] = \Pr[\pi_1 \wedge \pi_2] = \frac{1}{q_u}$.

If A_1 wins Game 1 with an advantage of ε, then B has a probability greater than $\frac{\varepsilon}{q_o}$ to determine whether $X = abP$. □

Lemma 4. *Assuming that adversary A_2 can win Game 4, then algorithm B can be constructed to solve the ECDDDH problem.*

Proof. Suppose that the tuple (P, aP, bP, X) is an example of an ECDDDH problem. To determine whether $X = abP$, B will play the part of the challenger.

Setup: B runs the setup (1^k) program to obtain the public parameters $\Omega = \{F_q, E/F_q, G_q, P, G, H_0, H_1, H_2, P_{pub}\}$, where $msk = \{x\}$ and $P_{pub} = xP$. Then, it randomly selects $x_u, x_C, r_C \in Z_q^*$, and sets

$$P_u = x_u P, R_u = aP\ P_C = x_C P, R_C = r_C P,$$
$$e_C = r_C + xH_0(ID_C, R_C, P_C) \bmod q,$$
$$e_u = a + xH_0(ID_u, R_u, P_u) \bmod q$$

B sends Ω, PK_u, and (PK, SK) to A_2, while SK_u are unknown to A_2.

Phase 1: Execute the user's public key query before other queries using the identity ID_i. Set up multiple lists to store the queries and answers. All lists are initially empty.

User public key query: B keeps a list L_o of the tuple $(ID_i, x_i P, r_i P, r_i)$, and upon receiving an identity ID_i, performs the following steps.

Case 1. $ID_i = ID_i^\diamondsuit$, B randomly chooses a number $r_i^\diamondsuit \in Z_q^*$, setting $PK_i^\diamondsuit = (bP, r_i^\diamondsuit P)$, and adds the tuple $(ID_i^\diamondsuit, bP, \diamondsuit, r_i^\diamondsuit P, r_i^\diamondsuit)$ to the list L_o, where $\diamondsuit$ represents a null value.

Case 2. $ID_i \neq ID_i^\diamondsuit$, B randomly chooses two different numbers, $x_i, r_i \in Z_q^*$, setting $PK_i = (x_i P, r_i P)$, and adds the tuple $(ID_i, x_i P, r_i P, r_i)$ to the list L_o.

Replace-Public-Key query: Same as in Lemma 1.

Secret-Value query: B establishes a list L_s of tuple (ID_j, x_j). When A_2 asks the secret value of ID_j, if $ID_j = ID_i^\diamondsuit$, B fails and stops. Otherwise, B finds $(ID_j, x_j P, r_j P, r_j)$ in the list L_o, and returns x_j. If PK_j is replaced, B refuses to answer.

Partial-Private-Key query: Same as in Lemma 2.

Keyword Ciphertext Query: Same as in Lemma 1.

Keyword Trapdoor Query: Same as in Lemma 1.

Test Query: Same as in Lemma 1.

Challenge: A_2 submits a tuple $(W'_0, W'_1, \{ID_1{}^*, \cdots, ID_d{}^*\}, \{PK_1{}^*, \cdots, PK_d{}^*\})$, where $W'_0 = \{w'_{0,1}, w'_{0,2}, \cdots, w'_{0,l}\}$ and $W'_1 = \{w'_{1,1}, w'_{1,2}, \cdots, w'_{1,l}\}$ are challenging keywords not requested in the previous trapdoor and keyword ciphertext query. If $ID_i^\diamondsuit \notin \{ID_1{}^*, ID_2{}^*, \cdots,$

$ID_d{}^*\}$, B aborts. Otherwise, without losing generality, it is better to set ID_1^* as $ID_i^{\diamondsuit}$. B calculates $l_0{}^* = \sum_{i=1}^{d} H_0(ID_i^*, R_i^*, P_i^*)$. B picks $\xi \in \{0,1\}$ randomly, and computes

$$t = x_u \sum_{i=1}^{d} P_i{}^* + \sum_{i=2}^{d} x_i{}^* aP + l_u x \sum_{i=1}^{d} P_i{}^* + \sum_{i=1}^{d} r_i{}^* ap + x_u l_0{}^* P_{pub} + X$$

B selects an element $\eta_\xi \in_R Z_q^*$ and sets $T_{W'_{\xi,m+1}} = \eta_\xi P$, $T_{W'_{\xi,j}} = l^{-1} h_2(t)^j \sum_{r=1}^{l} h_1(w'_{\xi,j})^j P + \eta_\xi P_C$, where $0 \leqslant j \leqslant m$. Finally, B sends $T_{W'_\xi} = \{T_{w'_\xi,0}, T_{w'_\xi,1}, \cdots, T_{w'_\xi,m}, T_{w'_{\xi,m+1}}\}$ to adversary A_2.

Phase 2: Attacker A_2 can continue to execute various queries, but there is a limitation that attacker A_2 is not allowed to query the keyword ciphertext or trapdoor of W_0 or W_1.

Guess: A_2 returns ξ'.

Solve CDH problem: If $\xi' = \xi$, B returns 1, otherwise 0. If $X = abP$, then

$$\begin{aligned} t &= x_u \sum_{i=1}^{d} P_i{}^* + \sum_{i=2}^{d} x_i{}^* aP + l_u x \sum_{i=1}^{d} P_i{}^* + \sum_{i=1}^{d} r_i{}^* aP + x_u l_0{}^* P_{pub} + X \\ &= x_u P_0^* + \sum_{i=2}^{d} x_i{}^* aP + l_u x \sum_{i=1}^{d} P_i{}^* + x_u R_0^* + x_u l_0^* P_{pub} + abP \\ &= x_u P_0^* + \sum_{i=1}^{d} x_i{}^* aP + l_u x \sum_{i=1}^{d} P_i{}^* + x_u R_0^* + x_u l_0^* P_{pub} \\ &= x_u P_0^* + (a + l_u x) \sum_{i=1}^{d} P_i{}^* + x_u R_0^* + x_u l_0^* P_{pub} \\ &= (x_u + e_u) P_0^* + x_u R_0^* + x_u l_0^* P_{pub}. \end{aligned}$$

Therefore, $T_{W'_\xi}$ is a valid keyword ciphertext. Suppose that the advantage of A_2 winning the above game is ε. So,

$$\Pr[\xi' = \xi | X = abP] = \tfrac{1}{2} + \varepsilon.$$

If $X \neq abP$, then $T_{W'_\xi}$ is an invalid keyword ciphertext. A_2 has no advantage in distinguishing $\xi = 0$ from $\xi = 1$. Hence,

$$\Pr[\xi' = \xi | X = abP] = \tfrac{1}{2}.$$

Probability: Let q_o, q_r, and q_s be the number of user public key queries, Replace-Public-Key queries, and Secret-Value queries, respectively. The two events are as follows:

π_1: A_2 did not replace $ID_i^{\diamondsuit}$'s public key $P_i^{\diamondsuit}$ and queries the secret value for $ID_i^{\diamondsuit}$.

π_2: $ID_1^* = ID_i^{\diamondsuit}$.

It is not hard to obtain the following results.

$\Pr[\pi_1] = \frac{q_o - q_r - q_e}{q_u}$,

$\Pr[\pi_2 | \pi_1] = \frac{1}{q_o - q_r - q_e}$,

$\Pr[B\ success] = \Pr[\pi_1 \wedge \pi_2] = \frac{1}{q_u}$.

If A_2 wins Game 4 with an advantage of ε, then B has a probability greater than $\frac{\varepsilon}{q_o}$ to determine whether $X = abP$. □

Theorem 3. *Our CLVPFC-PEKS scheme is CKA-TIND safe in the standard model if the ECDDDH problem is hard.*

Proof. Theorem 3 holds from Lemma 3 and Lemma 4. □

Theorem 4. *Under the ECDLP assumption, it is not computationally feasible for the CSS to forge valid proof information through the result verification mechanism.*

Proof. The malicious CSS cannot forge a valid multi-signature on each returned record and pass the verification. Since it does not have the key of multiple data owners, it is computationally infeasible to forge a valid multi-signature. Therefore, the malicious CSS can only win the next security game by directly generating valid proof information

according to the wrong search result C^* instead of winning the next security game by forging multiple signatures. But, after the following analysis, this is also impossible.

Assume that the correct keyword ciphertext and its identity are $C = \{c_1, c_2, \cdots, c_n\}$ and $sig = \{sig_1, \cdots, sig_n\}$, where $sig_t = \{Y_t, V_t\}$. The malicious CSS may forge wrong proof information (ϕ_1^*, ϕ_2^*) on false search results $C^* = \{c_{k_1}^*, c_{k_2}^*, \cdots, c_{k_s}^*\}$, where

$$\phi_1{}^* = \sum_{\tau=1}^{s} H_1(c_{k_\tau}^*, id_{k_\tau}^*, Y_{k_\tau}, P_0),$$
$$\phi_2{}^* = \sum_{\tau=1}^{s} H_1(c_{k_\tau}^*, id_{k_\tau}^*, Y_{k_\tau}, Q_0).$$

If the forged proof information (ϕ_1^*, ϕ_2^*) can successfully pass the result verification mechanism, the malicious CSS will win the security game; otherwise, it will fail. Suppose a malicious CSS wins the game. We then know that

$$\sigma P = \sum_{\tau=1}^{s} Y_{k_\tau} + \phi_1^* P_0 + \phi_2^* Q_0 \tag{3}$$

The proof information of the correct keyword ciphertext C is (ϕ_1, ϕ_2), where

$$\phi_1 = \sum_{\tau=1}^{s} H_1(c_{k_\tau}, id_{k_\tau}, Y_{k_\tau}, P_0),$$
$$\phi_2 = \sum_{\tau=1}^{s} H_2(c_{k_\tau}, id_{k_\tau}, Y_{k_\tau}, Q_0).$$

The signature of the correct keyword ciphertext can pass the verification mechanism, so we have

$$\sigma P = \sum_{\tau=1}^{s} Y_{k_\tau} + \phi_1 P_0 + \phi_2 Q_0 \tag{4}$$

Subtract equation (3) from equation (4) to obtain

$$(\phi_1 - \phi_1{}^*) P_0 = (\phi_2{}^* - \phi_2) Q_0 \tag{5}$$

Because (ϕ_1, ϕ_2) is not equal to (ϕ_1^*, ϕ_2^*), then $\phi_1 \neq \phi_1^*$ or $\phi_2 \neq \phi_2^*$. Set $\Delta\phi_1 = \phi_1 - \phi_1{}^*$, $\Delta\phi_2 = \phi_2 - \phi_2{}^*$, then $\Delta\phi_1 \neq 0$ or $\Delta\phi_2 \neq 0$. Suppose $\Delta\phi_1$ is not zero, then $P_0 = \frac{\Delta\phi_2}{\Delta\phi_1} Q_0$. If the probability of $\Delta\phi_1 = 0$ is $\frac{1}{q}$, then the probability that we can break the ECDLP problem is $1 - \frac{1}{q}$, where q is the length of G_q. This means that if the malicious CSS can pass the verification, we can break the ECDLP problem. □

7. Performance Analysis

We will compare our scheme in depth with other certificateless-based or verifiable search schemes on computational complexity, storage overhead, and security.

7.1. Security Comparison

Table 3 details the comparison of our scheme with other schemes in terms of security, where RCCA denotes that the scheme resists the chosen ciphertext attack, ROFFKGA stands for resists the offline keyword guessing attack, RONKGA stands for resists the online keyword guessing attack, RIKGA denotes that the scheme protects against the insider keyword guessing attack, VER denotes that the scheme prevents malicious CSSs from returning incorrect search results, and PM denotes the model used for the proof. In Table 3, yes indicates that the scheme satisfies the property, no implies that it does not meet the property, unknown denotes that it is unknown (the scheme has neither been proven safe nor unsafe because of the lack of security proof), and "-" means that the scheme does not have the feature.

Table 3. Comparison of security of different schemes.

Scheme	RCCA	ROFFKGA	RIKGA	RONKGA	VER	PM
[12]	yes	no	no	no	unknown	SM
VCSE [13]	yes	no	no	no	yes	ROM
VCKSM [14]	unknown	yes	unknown	unknown	yes	SM
VMKS [16]	yes	no	unknown	unknown	yes	SM
VMKDO [15]	yes	yes	no	no	yes	SM
[19]	unknown	no	no	no	-	unknown
mCLPECK [17]	yes	unknown	no	no	-	ROM
ours	yes	yes	yes	yes	yes	SM

It is clear from Table 3 that our solution has significant security advantages. Specifically, Theorem 2 shows that the new scheme is resistant to the adaptive keyword selection attack in the standard model, i.e., the new scheme is ciphertext-indistinguishable. Theorem 3 shows that our scheme is secure against offline keyword-guessing attacks in the standard model. Theorem 4 shows that our scheme prevents malicious CSSs from returning incorrect search results. Wu et al. [55] constructed a PEKS scheme that can resist online keyword attacks and insider keyword guessing attacks based on the security of the Diffie–Hellman shared secret key. The new scheme's shared key $t = (x_0 + e_0)P_u + x_0R_u + x_0l_uP_{pub} = (x_u + d_u)P_0 + x_uR_0 + x_ul_0P_{pub}$ is only accessible to the DOs and retrievers. Therefore, any third party other than the retriever and the DO, including external attackers and malicious internal servers, cannot generate the correct keyword trapdoor and ciphertext. In other words, our scheme can resist online and insider keyword-guessing attacks.

Lu et al. [44] demonstrated that if an attacker can generate keyword ciphertext using the public keys of the CSS and the retriever, then they can perform an online keyword-guessing attack on PEKS schemes. Later, Shao et al. [47] pointed out that if the SCF-PEKS scheme can generate keyword ciphertext with only the public keys of the CSS, DO, and retriever, while the insider attacker (malicious CSS) can run both the keyword encryption algorithm and the test algorithm, then the insider attacker can try online keyword guessing attacks on SCF-PEKS scheme. Since the keyword ciphertexts in schemes [12,13,15,17,19] are all generated using public keys, all of them are insecure against insider keyword-guessing attacks and online keyword-guessing attacks.

PEKS schemes typically use keywords selected from a low-entropy keyword space. Therefore, based on this characteristic, offline keyword-guessing attacks can be launched naturally [41]. By performing this attack, any insider/outsider attacker can correctly guess the keywords in a given keyword trapdoor using a testing algorithm. Construct $E' = \sum_{i=1}^{t} H(W_i'')$, where W_i'' is the guessed keyword set. Verify that the equation $e(T_2, E'g + PK_{s.1}) = e(e(PK_{C,2}, T_1)^t, g)$ holds. If the above equation is equal, the keyword set W_i' contained in T_2 in the trapdoor is the same as the guessed keyword set W_i''. Therefore, the scheme in [13] is insecure under offline keyword attacks. According to a similar construction idea, the schemes in [12,13,16,19] are also insecure under offline keyword-guessing attacks.

From the above analysis and the security proof in Section 4, we can see that our scheme is resistant to offline keyword guessing attacks and keyword selection attacks in the standard model. Secondly, based on the security of the Diffie–Hellman shared secret key, our scheme is also secure against insider and online keyword-guessing attacks. In other words, of all the schemes, ours is the safest.

7.2. Computational Overhead Comparison

Next, we compare the computational complexity. To compare the computational complexity, we use the operation time of He et al.'s scheme [56] as the benchmark. He

et al. tested the time required for the relevant operations in the experimental environment of Samsung Galaxy S5 based on the Android 4.4.2 operating system, quad-core 2.45 G processor, and 2G byte memory. Table 4 shows the exact running time and symbols of the various operations. The mapping $e : G_1 \times G_1 \rightarrow G_2$ is a bilinear pair, where G_1 is an additive group of singular elliptic curves of order p defined on a finite field F_q, and G_2 is a multiplicative group of order p. The lengths of p and q are 512 bits and 160 bits, respectively. G is an additive group of non-singular elliptic curves of order q defined on the prime finite field F_q. The length of p and q is 160.

Table 4. Symbol definition.

Symbols	Definition
T_{bp}	Running time required for a bilinear pairing operation, $T_{bp} \approx 32.713$ ms
T_{htp}	Running time required for a hash-to-operation, $T_{htp} \approx 33.582$ ms
T_{sm}	Running time required for a scalar multiplication operation in G_1, $T_{sm} \approx 13.405$ ms
$T_{\exp}$	Running time required for an exponentiation operation in G_2, $T_{\exp} \approx 2.249$ ms
$T'sm$	Running time required for a scalar multiplication operation in G, $T_{sm} \approx 3.335$ ms
u	Number of data users
d	Number of data owners
m	Number of keywords in index
l	Number of keywords contained in trapdoor
n	Number of ciphertexts
s	Number of search keywords

Because no SE scheme requires pairing and supports conjunctive keywords, we selected five SE schemes with similar functionalities for comparison. The results are shown in Table 5. Because each scheme has different settings, making direct horizontal comparisons is difficult. To ensure objectivity, we have established uniform parameter standards for comparison. The details are as follows: we assume that there are d data owners, u data consumers, n ciphertexts, m keywords contained in the keyword ciphertexts, l keywords contained in the trapdoors, and s keywords contained in the keyword ciphertexts obtained after the query. For algorithms that are not involved, we mark them with "-" in Table 5.

Table 5. Comparison of calculation complexity of various schemes.

Scheme	KeyGen	Enc	Trap	Search	Verify
VCKSM [14]	$(u+d+1)T_{sm}$	$(2nd + n(m+2))T_{sm} + 2nT_{\exp} + 2nT_{bp} + nT_{htp}$	$3T_{sm}$	$(n+3)T_{sm} + (n(m+1)+1)T_{bp}$	$(2s+1)T_{sm} + sT_{htp} + 2T_{bp}$
VMKS [16]	$(5u+5d)T_{sm} + (u+d)T_{htp}$	$nT_{htp} + (6n+mn)T_{sm}$	$(s+5)T_{sm}$	$4nT_{bp}$	$(s+2)T_{sm} + sT_{htp} + 3T_{bp}$
VCSE [13]	$(2u+1)T_{sm}$	$n(m+2)T_{\exp} + 3nT_{sm} + 2nT_{bp}$	$T_{sm} + T_{\exp}$	$2nT_{sm} + 3nT_{\exp} + 4nT_{bp}$	$2sT_{sm} + 3sT_{\exp} + 4sT_{bp}$
mCLPECK [17]	$2uT_{sm}$	$2unT_{sm} + 2nmT_{\exp} + 2nmT_{htp}$	$2sT_{htp} + 3T_{sm}$	$3nT_{bp}$	—
VMKDO [15]	$(d+1)T_{sm}$	$(3nm + 2n + 3)T_{sm} + 3nT_{\exp} + nT_{htp} + 3T_{bp}$	$2T_{sm}$	$2nT_{sm} + 2nT_{bp} + nT_{\exp}$	$(2s+1)T_{sm} + 2T_{bp} + sT_{htp}$
ours	$(2d+2u+1)T'$sm	$[nd + n(m+4)]T'sm$	$6T'sm$	$(1+n+nm)T'sm$	$3T'sm$

The data presented in Table 5 are theoretical calculations, but to more accurately evaluate the actual performance of our solution, we also need to simulate real-world scenarios. In the real world, there is a larger dataset and more participants. Therefore, it is worthwhile to set the number of ciphertexts from 1 to 100,000 ($n \in [1, 100000]$), the number of data owners (d), the number of keywords contained in trapdoors (l), and the number of data ciphertexts contained in query results (s) from 1 to 10,000.

The KeyGen algorithm's computational overhead for VCKSM [14], VMKS [16], and our scheme is affected by d and u, while the computational overhead of VCSE [13] is only affected by d. This is because the VCSE scheme's signing is performed with the system's private key and does not utilize the user's private key. Additionally, the computational overhead of VMKDO [15] is only influenced by u, as the keyword encryption of VCSE [13] is performed using the CSS's public key and not the user's public key. For comparison, assume $d = u$. Figure 3a illustrates that VMKS [16] has a much higher computational overhead in KeyGen compared to the other schemes. Our scheme and VMKDO [15] have roughly equal computational overhead in KeyGen and outperform the others.

In Figure 3b, we evaluate the computational burden of the Enc algorithm in these schemes by varying the number of ciphertexts from 1 to 100,000 ($n \in [1, 100000]$), assuming $u = d = 100$ and $m = n$. The computational overhead of the Enc algorithm for all the schemes almost increases with the number of n. Note that the mCLPECK [17] scheme has a much higher computational overhead in the Enc algorithm than the other schemes; our scheme and the VCSM [13] scheme have roughly equal computational overheads in the Enc algorithm, and both outperform the others.

In the Trap algorithm, the computational overhead of the VCKSM [14] scheme with the mCLPECK [17] scheme is affected by l, and both grow linearly with l, whereas the computational overheads of the other schemes are fixed values. In Figure 3c, we evaluate the computational burden of the Trap algorithm by varying the number of keywords included in the trapdoor from 1 to 10,000 ($l \in [1, 10000]$). Note that the computational overheads of the VMKS scheme and the mCLPECK [17] scheme in the Trap algorithm are much higher than those of the other schemes; the computational overheads of our scheme and the VCSM [13] scheme in the Trap algorithm are roughly equal, and both outperform the others.

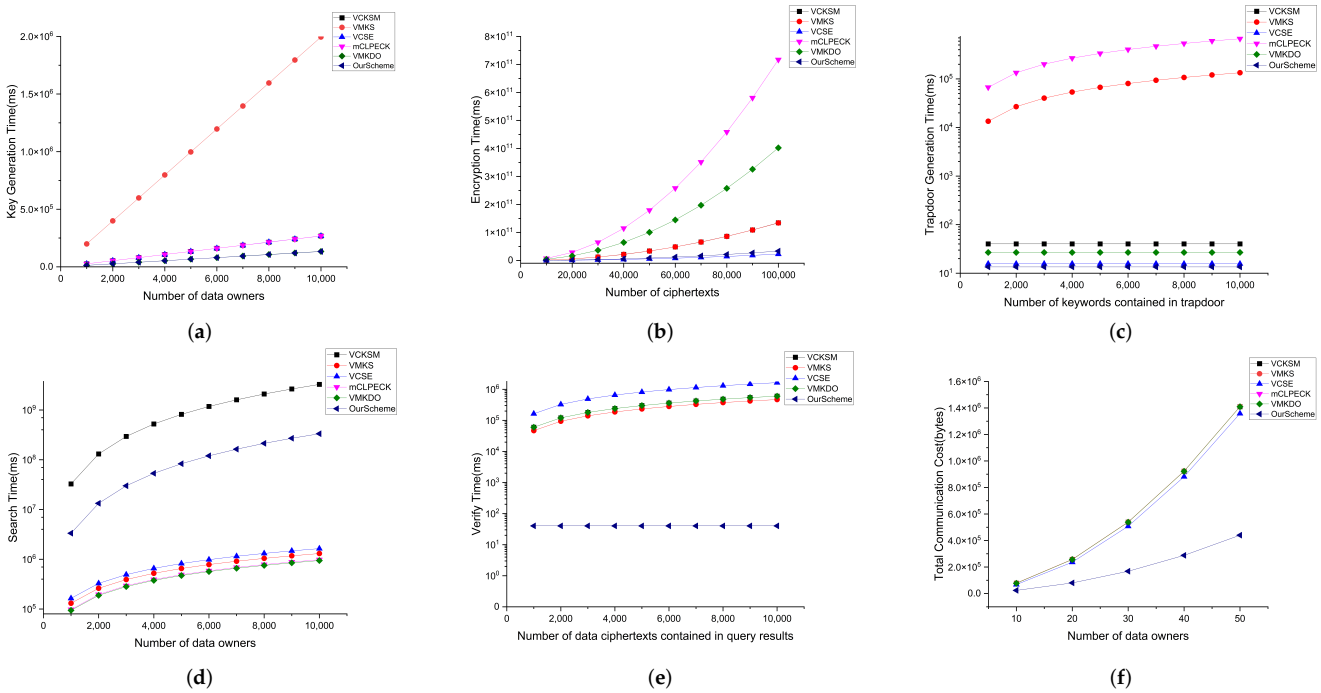

Figure 3. Comparison results. (**a**) Computational overhead of the KeyGen algorithm. (**b**) Computational overhead of the Enc algorithm. (**c**) Computational overhead of the Trap algorithm. (**d**) Computational overhead of the Search algorithm. (**e**) Computational overhead of the Verify algorithm. (**f**) Communication overheads.

In the Search algorithm, the computational overhead of the VCKSM [14] scheme and our scheme is affected by n and m, while the other schemes are only affected by n. In Figure 3d, for ease of discussion, we assume $m = n$ and evaluate the computa-

tional burden of the Search algorithm by varying the number of data owners from 1 to 10,000 ($d \in [1, 10,000]$). It can be seen that the computational overhead of all schemes almost always increases with n. Note that the computational overhead of the VCKSM [14] scheme with our scheme in the Search algorithm is slightly higher than that of the other schemes, which are roughly equal.

In the Verify algorithm, the computational overhead of all schemes except ours is affected by s. The computational overhead of our scheme is a fixed value, and the other schemes grow linearly with s. In Figure 3e, we evaluate the computational overhead of the Verify algorithm by varying the number of data ciphers contained in the query results from 1 to 10,000 ($s \in [1, 10, 000]$). Note that the computational overhead of the VCSE [13] scheme in Verify's algorithm is higher than that of the other schemes, while the computational overhead of our scheme is fixed. As l1 becomes larger, our scheme is more advantageous.

To sum up, except for the Search algorithm, our scheme has a lower storage cost in the KeyGen, Trap, and Verify algorithms than the other five schemes, among which our scheme has a much lower storage cost in the Verify algorithm than the other schemes. Considering the storage costs of all algorithms, our solution has the lowest storage cost of all solutions.

7.3. Storage Cost Comparison

Below, we compare storage costs. Let $|G_1|$, $|G_2|$, $|G|$, and $|Z_q|$ represent the sizes of elements in G_1, G_2, G, and Z_q, respectively. Then $|G_1| = G_2$= 512 bytes, $|G|$ = 160 bytes, and $|Z_q|$ = 160 bytes. If the conjunctive keyword-searchable encryption scheme has no verification function, fewer signatures for the keyword ciphertext will be stored. Therefore, we select only verifiable conjunctive-keyword-searchable encryption schemes to compare the size of storage costs. Table 6 illustrates the comparison results.

Table 6. Storage cost comparison.

Scheme	KeyGen	Trap	Search	Verify
VCKSM [14]	$(d+u+1)(\|Z_p\|+\|G_1\|)+\|G_1\|$	$(m+2)\|G_1\|+2\|Z_p\|$	$(m+3)\|G_2\|+\|G_1\|$	$(s+1)\|Z_p\|+\|G_1\|+2\|G_2\|$
VMKS [16]	$2(u+d)(\|Z_p\|+\|G_1\|)$	$(l+5)\|G_1\|+l\|Z_p\|$	$(l+2)\|G_1\|+4\|G_2\|$	$(s+2)\|G_1\|+(s+1)\|Z_p\|+3\|G_2\|$
VCSE [13]	$(u+1)(\|Z_p\|+\|G_1\|+\|G_2\|)$	$(l+1)(\|Z_p\|+\|G_1\|)+\|G_1\|$	$8\|G_2\|+2\|G_1\|$	$2\|Z_p\|+(2s+3)\|G_1\|+2\|G_2\|$
VMKDO [15]	$(2d+1)\|G_1\|+(d+1)\|Z_p\|$	$\|G_1\|+(l+1)\|Z_p\|$	$2\|G_1\|+3\|G_2\|$	$\|Z_p\|+(2s+1)\|G_1\|)+2\|G_2\|$
our	$(u+d+1)(\|Z_p\|+\|G\|)$	$(m+6)\|G\|$	$(m+2)\|G\|$	$3\|Z_p\|+6\|G\|$

In the KeyGen algorithm, the storage costs of VCKSM [14], VMKS [16], and our scheme are all affected by d and u, while the storage cost of the VCSE scheme is only affected by d and the storage cost of VMKDO [15] is only affected by u. For the sake of discussion, we assume $d = u$. So, the storage cost for all scenarios grows linearly with u. Since $|Z_q| = |G| = \frac{1}{2}|G_1|, |G_2| = |G_1|$, our solution has the lowest storage cost of all.

In the Trap algorithm, the storage cost of VCKSM [14] and our scheme is affected by m1, while the storage cost of VMKS [16], VCSE [13], and VMKDO [15] is affected by l. For the sake of discussion, we assume $l = m$. Then, all the above schemes increase linearly with m. Since $|Z_q| = |G| = \frac{1}{2}|G_1|, |G_2| = |G_1|$, our solution, VCSE [13], and VMKDO [15] are approximately the same and much lower than VCKSM [14] and VMKS [16].

In the Search algorithm, the storage cost of VCKSM [14] and our scheme is affected by m, the storage cost of VMKS [16] is only affected by l, and the storage cost of VCSE [13] and VMKDO [15] is a fixed value. For the sake of discussion, we assume $l = m$. Then, VCKSM [14], VMKS [16], and our scheme increase linearly with l. Because $|Z_q| = |G| = \frac{1}{2}|G_1|, |G_2| = |G_1|$, VCKSM [14], and VMKS [16] have the highest storage costs, our solution is second, and VCSE [13] and VMKDO [15] have the lowest storage cost.

In the Search algorithm, the storage costs of VCKSM [14] and our scheme are affected by m, the storage cost of VMKS [16] is only affected by l, and the storage costs of VCSE [13] and VMKDO [15] are fixed. For the sake of discussion, let us assume $l = m$. Then, VCKSM [14], VMKS [16], and our scheme all increase linearly with l. Because $|Z_q| = |G| =$

$\frac{1}{2}|G_1|, |G_2| = |G_1|$, VCKSM [14] and VMKS [16] have the highest storage costs, our solution is second, and VCSE [13] and VMKDO [15] have the lowest storage cost. In the Verify algorithm, the storage cost of all schemes except ours is affected by s. Therefore, the storage cost of their solutions increases linearly with s, and the storage cost of our solution is a fixed value. As a result, our solution has significantly lower storage costs than others.

To sum up, except for the Search algorithm, our scheme has a lower storage cost in all the algorithms (KeyGen, Trap, and Verify) than the other five schemes, among which our scheme has a much lower storage cost in the Verify algorithm than the other schemes. Considering the storage costs of all algorithms, our solution has the lowest storage cost of all solutions.

7.4. Comparison of Communication Costs

Finally, we compare the communication costs. If the conjunctive keyword-searchable encryption scheme has no verification function, the communication cost does not need to consider the signature communication cost in the communication cost. Therefore, we only select the conjunctive keyword-searchable encryption schemes that support verifiability for comparison. Table 7 shows the comparison results.

Table 7. Comparison of communication costs.

Scheme	Index Size	Signature Size	Trapdoor Size	Total Size
VCKSM [14]	$n(m+2)\|G_1\| + n\|G_2\|$	$n\|G_1\|$	$(m+2)\|G_1\| + \|Z_q^*\|$	$n(m+3)\|G_1\| + (m+2)\|G_1\| + n\|G_2\| + \|Z_q^*\|$
VMKS [16]	$n(m+4)\|G_1\|$	$n\|G_1\|$	$5\|G_1\| + \|Z_q^*\|$	$n(m+5)\|G_1\| + 5\|G_1\| + \|Z_q^*\|$
VCSE [13]	$n(m+2)\|G_2\|$	$n\|G_1\|$	$\|G_1\| + \|G_2\| + \|Z_q^*\|$	$n(m+2)\|G_2\| + (n+1)\|G_1\| + \|G_2\| + \|Z_q^*\|$
VMKDO [15]	$n(m+2)\|G_1\| + 2n\|G_2\|$	$n\|G_1\|$	$\|G_1\| + 2\|Z_q^*\|$	$n(m+3)\|G_1\|) + 2n\|G_2\| + \|G_1\| + 2\|Z_q^*\|$
ours	$n(m+3)\|G\|$	$n\|G\|$	$(m+2)\|G\|$	$n(m+4)\|G\| + (m+2)\|G\|$

As shown in Table 7, our scheme has the shortest signature length. The signature lengths of the other schemes are the same. VMKS [16] and VMKDO [15] have the longest index lengths because they are at least 5 bytes larger than the other schemes. The index lengths of the others are roughly the same. The trapdoor lengths of VCKSM [14] are the longest, followed by our scheme, and the shortest is of the VMKDO [15] scheme. Assuming $m = n$, we compare the total communication cost of these schemes in Figure 3(f). According to Figure 3f, the total communication cost of our scheme is lower than the total communication cost of all schemes.

Overall, our solution is more efficient than others in computation, storage, and communication. Most importantly, it excels in security.

8. Conclusions

This article proposes a novel certificateless, verifiable, bilinear, pair-free, conjunctive keyword search encryption scheme (CLVPFC-PEKS), aiming to provide a secure and efficient data search method for the Internet of Things healthcare (IoMT) field. Our solution solves the problems in existing public key searchable encryption technologies, such as low computational efficiency and susceptibility to keyword guessing attacks. We have demonstrated under the standard model that the CLVPFC-PEKS scheme can resist both choosen keyword and ciphertext attacks and offline keyword guessing attacks, and based on the security of the Diffie–Hellman shared secret key to demonstrate that the CLVPFC-PEKS scheme can resist online/internal keyword guessing attacks. In addition, we also conducted a detailed analysis of the performance of the scheme. The results indicate that our scheme has significant advantages in terms of security, computational complexity, storage overhead, and communication costs compared to existing schemes. Overall, our solution provides a new solution for secure and efficient data search in the medical Internet of Things, meeting the urgent needs for data security and privacy protection.

Author Contributions: Methodology, W.L.; Formal analysis, W.L.; Data curation, Y.W. and H.Z.; Writing—original draft, W.L.; Writing—review & editing, W.L.; Visualization, Y.W. and Y.G.; Supervision, J.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This paper was supported by the National Natural Science Foundation of China (No.61962011); the Guiyang City Science and Technology Plan Project (No.[2021]43-8); Guizhou Normal University Academic New Seedling Fund Project (QianShiXinMiao[2021]B09).

Data Availability Statement: Data are contained within the article.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Iera, L.A.; Morabito, G. The Internet of Things: A survey. *Comput. Netw.* **2010**, *54*, 2787–2805. 2010.05.010. [CrossRef]
2. Zanella, A.; Bui, N.; Castellani, A.; Vangelista, L.; Zorzi, M. Internet of Things for Smart Cities. *IEEE Internet Things J.* **2014**, *1*, 22–32. [CrossRef]
3. Bellavista, P.; Cardone, G.; Corradi, A.; Foschini, L. Convergence of MANET and WSN in IoT Urban Scenarios. *IEEE Sens. J.* **2013**, *13*, 3558–3567. [CrossRef]
4. He, D.J.; Ye, R.; Chan, S.; Guizani, M.; Xu, Y.P. Privacy in the Internet of Things for Smart Healthcare. *IEEE Commun. Mag.* **2018**, *56*, 38–44. [CrossRef]
5. Chen, Y.; Lu, J.; Jan, J. A Secure EHR System Based on Hybrid Clouds. *J. Med. Syst.* **2012**, *36*, 3375–3384. [CrossRef]
6. Jagadeeswari, V.; Subramaniyaswamy, V.; Logesh, R.; Vijayakumar, V. A study on medical Internet of Things and Big Data in personalized healthcare system. *Health Inf. Sci. Syst.* **2018**, *6*, 14. [CrossRef] [PubMed]
7. Wagner, D.X.S.; Perrig, A. Practical Techniques for Searches on Encrypted Data. In Proceedings of the IEEE Symposium on Security & Privacy, Berkeley, CA, USA, 12–15 May 2002; pp. 44–55.
8. Li, M.; Yu, S.; Ren, K.; Lou, W. Securing personal health records in cloud computing: Patient-centric and fine-grained data access control in multi-owner settings. In Proceedings of the ICST Conference Security and Privacy in Communication Networks PP, Singapore, 7–9 September 2010; pp. 89–106. [CrossRef]
9. Wang, B.Y.; Li, H.; Liu, X.F.; Li, X.Q.; Li, F.H. Preserving identity privacy on multi-owner cloud data during public verification. *Secur. Commun. Netw.* **2014**, *7*, 2104–2113. [CrossRef]
10. Miao, Y.; Liu, X.; Raymond Choo, K.; Deng, R.H.; Li, J.; Li, H.; Ma, J. Privacy-Preserving Attribute-Based Keyword Search in Shared Multi-owner Setting. *IEEE Depend. Secur.* **2021**, *18*, 1080–1094. [CrossRef]
11. Padhya, M.; Jinwala, D.C. CRSQ-KASE: Key Aggregate Searchable Encryption Supporting Conjunctive Range and Sort Query on Multi-owner Encrypted Data. *Arab. J. Sci. Eng.* **2020**, *45*, 3133–3155. [CrossRef]
12. Hwang, M.; Hsu, H.T.; Lee, C.C. A New Public Key Encryption with Conjunctive Field Keyword Search Scheme. *Inf. Technol. Control.* **2014**, *43*, 277–288. [CrossRef]
13. Miao, Y.; Ma, J.; Wei, F.; Liu, Z.; Wang, X.A.; Lu, C. VCSE: Verifiable conjunctive keywords search over encrypted data without secure-channel. *Peer Netw.* **2017**, *23*, 995–1007. [CrossRef]
14. Miao, Y.; Ma, J.; Liu, X.; Jiang, Q.; Zhang, J.; Shen, L.; Liu, Z. VCKSM: Verifiable conjunctive keyword search over mobile e-health cloud in shared multi-owner settings. *Pervasive Mob. Comput.* **2017**, *40*, 205–219. [CrossRef]
15. Miao, Y.; Ma, J.; Liu, X.; Liu, Z.; Wei, F. VMKDO: Verifiable multi-keyword search over encrypted cloud data for dynamic data-owner. *Peer Netw. Appl.* **2018**, *11*, 287–297. [CrossRef]
16. Miao, Y.; Weng, J.; Liu, X.; Choo, K.R.; Liu, Z.; Li, H. Enabling verifiable multiple keywords search over encrypted cloud data. *Inform. Sci.* **2018**, *465*, 21–37. [CrossRef]
17. Fan, M.M.M.Q.; Feng, D.G. Multi-user certificateless public key encryption with conjunctive keyword search for cloud-based telemedicine. *J. Inf. Secur. Appl.* **2020**, *55*, 102652. [CrossRef]
18. Wu, L.; Zhang, Y.; Ma, M.; Kumar, N.; He, D. Certificateless searchable public key authenticated encryption with designated tester for cloud-assisted medical Internet of Things. *Ann. Telecommun.* **2019**, *74*, 423–434. [CrossRef]
19. Wang, Y.L.; Li, J.G. Keyword guessing attacks on a public key encryption with keyword search scheme without random oracle and its improvement. *Inform. Sci.* **2019**, *479*, 270–276. [CrossRef]
20. Liu, X.; Sun, Y.; Dong, H. A pairing-free certificateless searchable public key encryption scheme for IoMT. *J. Syst. Architect.* **2023**, *139*, 102885. [CrossRef]
21. Senouci, M.R.; Benkhaddra, I.; Senouci, A.; Li, F.G. A provably secure free-pairing certificateless searchable encryption scheme. *Telecommun. Syst.* **2022**, *80*, 383–395. [CrossRef]
22. Hu, Z.Y.; Deng, L.Z.; Wu, Y.Y.; Shi, H.Y.; Gao, Y. Secure and Efficient Certificateless Searchable Authenticated Encryption Scheme without Random Oracle for Industrial Internet of Things. *IEEE Syst. J.* **2023**, *17*, 1304–1315. [CrossRef]
23. Boneh, D.; Crescenzo, G.D.; Ostrovsky, R.; Persiano, G. Public Key Encryption with Keyword Search. In *Advances in Cryptology*; Springer: Berlin/Heidelberg, Germany 2004; pp. 506–522._30. [CrossRef]

24. Safiavinaini, J.B.; Susilo, W. Public Key Encryption with Keyword Search Revisited. In Proceedings of the International Conference on Computational Science and Its Applications (ICCSA 2008), Perugia, Italy, 30 June–3 July 2008; pp. 1249–1259.
25. Rhee, H.S.; Park, J.H.; Susilo, W.; Lee, D.H. Improved searchable public key encryption with designated tester. In Proceedings of the International Symposium on Information, Seoul, Republic of Korea, 28 June–3 July 2009; p. 376.
26. Al-Riyami, S.S.; Paterson, K.G. Certificateless Public Key Cryptography. In *Advances in Cryptology—ASIACRYPT 2003*; Lecture Notes in Computer Science (LNCS, Volume 2894); Laih, C.S., Ed.; Springer: Berlin/Heidelberg, Germany, 2003; pp. 452–473.
27. Xu, P.; Jin, H.; Wu, Q.; Wang, W. Public-Key Encryption with Fuzzy Keyword Search: A Provably Secure Scheme under Keyword Guessing Attack. *IEEE Trans. Comput.* **2013**, *62*, 2266–2277. [CrossRef]
28. He, D.; Ma, M.; Zeadall, S.; Kumar, N.; Liang, K. Certificateless Public Key Authenticated Encryption with Keyword Search for Industrial Internet of Things. *IEEE Trans. Ind. Inform.* **2018**, *14*, 3618–3627. [CrossRef]
29. Ma, M.; He, D.; Khurram Khan, M.; Chen, J. Certificateless searchable public key encryption scheme for mobile healthcare system. *Comput. Electr. Eng.* **2018**, *65*, 413–424. [CrossRef]
30. Ma, M.; He, D.; Fan, S.; Feng, D. Certificateless searchable public key encryption scheme secure against keyword guessing attacks for smart healthcare. *J. Inf. Secur. Appl.* **2020**, *50*, 102429. [CrossRef]
31. Golle, P.; Stadon, J.; Waters, B. Secure Conjunctive Keyword Search over Encrypted Data. In *Applied Cryptography and Network Security* ; Jakobsson, M., Yung, M., Zhou, J., Ed.; Springer: Berlin/Heidelberg, Germany, 2004; pp. 31–45.
32. Hwang, Y.H.; Lee, P.J. Public Key Encryption with Conjunctive Keyword Search and Its Extension to a Multi-User System. In *Pairing-Based Cryptography—Pairing 2007*; Lecture Notes in Computer Science (LNSC,Volume 4575); Springer: Berlin/Heidelberg, Germany, 2007; p. 2.
33. Yang, Y.; Ma, M.D. Conjunctive Keyword Search With Designated Tester and Timing Enabled Proxy Re-Encryption Function for E-Health Clouds. *IEEE Inf. Foren. Sec.* **2016**, *11*, 746–759. [CrossRef]
34. Heng, S.H.; Kurosawa, K. K-Resilient Identity-Based Encryption in the Standard Model. In *Topics in Cryptology—CT-RSA 2004 2964*; Lecture Notes in Computer Science (LNCS, Volume 2964); Okamoto, T., Ed.; Springer: Berlin/Heidelberg, Germany, 2004; pp. 67–80.
35. Khader, D. Public Key Encryption with Keyword Search Based on K-Resilient IBE. In *Computational Science and Its Applications—ICCSA 2006*; Lecture Notes in Computer Science (LNTCS, Volume 3982); Gavrilova, M., Ed.; Springer: Berlin/Heidelberg, Germany, 2006; pp. 298–308.
36. Xu, H.M.Y.X.; Zhao, H.T. An Efficient Public Key Encryption with Keyword Scheme Not Using Pairing. In Proceedings of the First International Conference on Instrumentation, Nagpur, Maharashtra, India, 21–22 April 2011.
37. Vallent, T.F.; Kim, H. A Pairing-Free Public Key Encryption with Keyword Searching for Cloud Storage Services. In *e-Infrastructure and e-Services for Developing Countries*; Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering (LNICST, Volume 135); Springer: Berlin/Heidelberg, Germany, 2014; p. 70.
38. Yang, N.; Xu, S.; Quan, Z. An Efficient Public Key Searchable Encryption Scheme for Mobile Smart Terminal. *IEEE Access* **2020**, *8*, 77940–77950. [CrossRef]
39. Lu, Y.; Li, J. Constructing pairing-free certificateless public key encryption with keyword search. *Front. Inf. Technol. Electron. Eng.* **2019**, *20*, 1049–1060. [CrossRef]
40. Ma, M.; Luo, M.; Fan, S.; Feng, D. An Efficient Pairing-Free Certificateless Searchable Public Key Encryption for Cloud-Based IIoT. *Wirel. Commun. Mob. Com.* **2020**, *2020*, 8850520. [CrossRef]
41. Byun, J.W.; Rhee, H.S.; Park, H.A.; Lee, D.H. Off-line keyword guessing attacks on recent keyword search schemes over encrypted data. In *Secure Data Management*; Lecture Notes in Computer Science (LNISA, Volume 4165); Springer: Berlin/Heidelberg, Germany, Jonker, W., Petkovic, M., Eds.; 2006; pp. 75–83.
42. Lin, X.J.; Sun, L.; Qu, H.P.; Liu, D.X. On the Security of Secure Server-Designation Public Key Encryption with Keyword Search. *Comput. J.* **2018**, *61*, 1791–1793. [CrossRef]
43. Huang, Q.; Li, H.B. An efficient public-key searchable encryption scheme secure against inside keyword guessing attacks. *Inform. Sci.* **2017**, *403*, 1–14. [CrossRef]
44. Wang, Y.L.; Li, J. On Security of a Secure Channel Free Public Key Encryption with Conjunctive Field Keyword Search Scheme. *Inf. Technol. Control* **2018**, *47*, 56–62. [CrossRef]
45. Jeong, I.R.; Kwon, J.O.; Hong, D.; Lee, D.H. Constructing PEKS schemes secure against keyword guessing attacks is possible? *Comput. Commun.* **2009**, *32*, 394–396. [CrossRef]
46. Wang, B.J.; Chen, T.H.; Jeng, F.G. Security Improvement against Malicious Servers in dPEKS Scheme. *Int. J. Inf. Educ. Technol.* **2011**, *1*, 4.
47. Shao, Z.Y.; Yang, B. On security against the server in designated tester public key encryption with keyword search. *Inform. Process. Lett.* **2015**, *115*, 957–961. [CrossRef]
48. Chai, Q.; Gong, G. Verifiable symmetric searchable encryption for semi-honest-but-curious cloud servers. In Proceedings of the IEEE International Conference on Communications, Ottawa, ON, Canada, 10–15 June 2012.
49. Sun, W.; Liu, X.; Lou, W.; Hou, Y.T.; Li, H. Catch You If You Lie to Me: Efficient Verifiable Conjunctive Keyword Search over Large Dynamic Encrypted Cloud Data. In Proceedings of the 34th IEEE Conference on Computer Communications (INFOCOM), Hong Kong, China, 26 April–1 May 2015.
50. Ismail, M.; Abd El-Gawad, A.F. Revisiting Zero-Trust Security for Internet of Things. *Sustain. Mach. Intell. J.* **2023**, *3*. [CrossRef]

51. Alenizi1, J.A.; Alrashdi, I. SFMR-SH: Secure Framework for Mitigating Ransomware Attacks in Smart Healthcare Using Blockchain Technology. *Sustain. Mach. Intell. J.* **2023**, *2*, 1–19. [CrossRef]
52. Hu, C.; Liu, P. An Enhanced Searchable Public Key Encryption Scheme with a Designated Tester and Its Extensions. *J. Comput.* **2012**, *7*, 716–723. [CrossRef]
53. Rhee, H.S.; Park, J.H.; Susilo, W.; Lee, D.H. Trapdoor security in a searchable public-key encryption scheme with a designated tester. *Syst. Softw.* **2010**, *83*, 763–771. [CrossRef]
54. Fang, L.; Susilo, W.; Ge, C.; Wang, J. Public key encryption with keyword search secure against keyword guessing attacks without random oracle. *Inform. Sci.* **2013**, *238*, 221–241. [CrossRef]
55. Wu, L.; Chen, B.; Zeadally, S.; He, D. An efficient and secure searchable public key encryption scheme with privacy protection for cloud storage. *Soft Comput.* **2018**, *22*, 7685–7696. [CrossRef]
56. He, D.B.; Wang, H.Q.; Wang, L.N.; Shen, J.; Yang, X.Z. Efficient certificateless anonymous multi-receiver encryption scheme for mobile devices. *Soft Comput.* **2017**, *21*, 6801–6810. [CrossRef]